UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

PROPUESTA DE UN MODELO FORMAL PARA MÁQUINAS SOCIALES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

TOMÁS CARRASCO ESCAFF

PROFESOR GUÍA:
CLAUDIO GUTIÉRREZ GALLARDO

MIEMBROS DE LA COMISIÓN:
ALEJANDRO HEVIA ANGULO
ALEX BORQUEZ GRIMALDI

SANTIAGO DE CHILE
2014

Resumen

El presente trabajo tiene por objetivo proponer una formalización matemática para el concepto de máquinas sociales. Este concepto fue introducido a fines del siglo pasado para referirse al rol que los computadores deben jugar en la sociedad. En este sentido, las interacciones humanas deben poder darse en línea con tanta facilidad como en persona. El poder de los computadores que soportan estas interacciones en línea y que procesan los datos involucrados en ellas, debe ser aprovechado para liberar a las personas de las tareas burocráticas y dejarles en cambio los aspectos más creativos e intuitivos que intervienen en la socialización. Con el tiempo, este concepto inicial de máquinas sociales fue ampliando su significado. Actualmente abarca en forma global el estudio de las interrogantes que surgen cuando los elementos sociales y computacionales se mezclan en el software. Esta noción hoy forma parte de un conjunto más amplio de ideas académicas similares entre sí, que estudian desde distintos enfoques temas en común, como lo son el impacto del avance de la computación en el potencial intelectual colectivo de las multitudes, la posibilidad de articular nuevas formas de acción colectiva, y las relaciones de producción y consumo en la Web, entre otros. Hasta el momento no se ha propuesto una formalización matemática del concepto de máquinas sociales. Con una herramienta así se podría describir con rigurosidad los patrones de interacción y computación presentes en las máquinas sociales, lo que permitiría mejorar el análisis, clasificación y desarrollo de estos sistemas.

Para alcanzar el objetivo propuesto, este trabajo se dividió en dos partes. En la primera parte se realizó una revisión bibliográfica de la noción de máquina social. Se comenzó explorando la noción original de máquinas sociales, se continuó estudiando los principales conceptos académicos que la rodean, y se finalizó analizando los principales enfoques de investigación en máquinas sociales hasta la fecha. En la segunda parte se modeló formalmente la noción de máquinas sociales. Para ello se comenzó estableciendo las nociones preliminares necesarias para la comprensión del modelo, y luego se desarrolló el modelo en tres etapas, formalizando en cada etapa lo que se denominó comportamiento interno, comportamiento persistente y comportamiento externo, respectivamente. El modelo de máquinas sociales desarrollado se construyó en base al formalismo de máquinas de Turing, en base a la noción de persistencia entre computaciones presente en el modelo de máquinas de Turing persistentes, y en base al formalismo del π-cálculo que permite modelar comunicación entre distintos actores. La formalización finalizó evidenciando cómo el modelo desarrollado permite describir características prácticas de máquinas sociales reales, mediante la construcción de expresiones formales que las modelan. Una de las principales líneas de trabajo futuro que deja esta memoria dice relación con la posibilidad de modelar las maquinas sociales usando sólo parte de los formalismos base utilizados, desafío complejo de afrontar por su naturaleza metateórica.

Recuerdo que eran días muy fríos,

y tú me abrazaste con tu calor y me cuidaste.

A todos aquellos que me acompañaron en los tiempos más difíciles.

Agradecimientos

Agradezco a mi Mamá y su esfuerzo inclaudicable por entregarme una buena educación.

Agradezco a mi Tata y a mi Abuelita.

Agradezco a Lili, Claudio e Ingrid.

Agradezco a mi familia, que me apoyó en todo momento.

Agradezco especialmente a mi Tío Roberto.

Agradezco a mi Tía Lili, nuevamente.

Agradezco a mis primos.

Agradezco a la Susi.

Agradezco a mis amigos.

Al Max, al Shisher, a la Chica, al Edú, a la Cami, al Pizarro, a la Marité y al Chigo.

A la Kati, al Fallo, al Jejo, al Danilo, y a la Coni.

Agradezco a los chiquillos del DCC.

Y a todos los que se me quedan en el tintero.

...

¡Muchas gracias a todos ustedes!

Agradezco a Fondecyt No. 1110287.

Tabla de contenido

Índice de tablas

Índice de figuras

Introducción

La evolución de la Web ha permitido que gran parte de las formas de socialización que tradicionalmente se desarrollaban de forma presencial hoy puedan darse en línea. Esta transformación del espacio donde ocurren las interacciones sociales ha sido objeto de estudio de múltiples disciplinas, siendo la computación una de ellas. En lo que respecta a esta ciencia, distintas visiones y conceptos se han generado en respuesta a este cambio social. Entre ellos se encuentra la noción de *máquinas sociales*. Bajo esta noción se estudia la forma en que los elementos sociales y computacionales se entremezclan en el software. En sus orígenes, esta noción fue introducida para destacar el rol que los computadores debían tener en una sociedad que podía reflejar con mayor nitidez sus procesos sociales en el espacio Web. Si las interacciones se dan en línea, entonces se puede usar el poder de cómputo propio de las máquinas que dan soporte a este tipo de interacciones para realizar las tareas de administración y cálculo que componen los procesos sociales, dejando así a las personas las tareas creativas y que requieran de la intuición humana. Estas máquinas de cómputo que asisten los procesos sociales dejando a las personas el quehacer creativo y así mismas el quehacer administrativo corresponde a la noción original de máquina social y que Tim Berners-Lee introdujo en 1999 en su libro *«Weaving the Web»*.

Pronto el estudio de esta noción se vería acompañado de otras ideas y conceptos con que se estudiaba también la relación entre los aspectos sociales y computacionales. La Web 2.0, el software social, la noción de *prosumo* en la Web, el *crowdsourcing* y los conceptos que estudian el potencial intelectual colectivo y de organización sin precedentes que las multitudes están exhibiendo en los sistemas Web son algunos de los integrantes del escenario donde habita la noción de máquinas sociales. El estudio académico de máquinas sociales se alimentaría de las investigaciones en estos terrenos vecinos, y los propios trabajos científicos sobre máquinas sociales se aproximarían de distinta forma a esta concepción inicial propuesta por Berners-Lee. Hoy día el área de investigación de máquinas sociales realiza un estudio coherente y cohesionado de las interrelaciones de los elementos sociales y computaciones en los sistemas Web, pero aún no se comparte una definición tan transversal como precisa entre los distintos académicos. Se han desarrollado muchos esfuerzos por obtener cada vez mejores caracterizaciones del fenómeno base de estudio, y se han logrado grandes avances. Mas no existe hasta la fecha un intento de precisar estas caracterizaciones en una noción formal. La elaboración de un modelo formal de máquinas sociales que permita modelar su comportamiento y representar formalmente las características de la fenomenología estudiada sería una gran contribución a esta área. Permitiría generar una primera propuesta para definir formalmente el objeto de estudio de esta área de investigación, propuesta sobre la cual podrían construirse otras posteriormente, de forma que el área fuera incorporando distintos elementos formales

para enfrentar mejor sus desafíos de investigación. Lo más importante es que si se desarrolla una noción formal de máquinas sociales que responda correctamente a la noción más intuitiva de máquinas sociales, entonces se estaría aportando significativamente en la comprensión de las formas en que se pueden interrelacionar los elementos sociales y computacionales en el software, lo que ayudaría a poner un ladrillo más en el intento de construir sistemas Web donde se aseguren propiedades de interacción social a través de métodos formales.

Esta memoria tiene por objetivo proponer una formalización matemática de la noción de máquinas sociales. Para ello, primero se explora junto al lector esta noción y luego se presenta el modelo formal de máquinas sociales. La exploración inicial considera el examen de la noción original de máquinas sociales, la revisión de una selección de los fenómenos que contextualizan este concepto, y el estudio de los principales enfoques adoptados por los trabajos de investigación académica sobre el tema. La formalización posterior presenta un modelo de máquinas sociales en tres etapas distintas. Primero se presenta la parte del modelo que describe el comportamiento interno de las máquinas sociales. Luego se presenta la parte del formalismo que modela la persistencia en el comportamiento de máquinas sociales. Finalmente se modela la máquina social como un actor más dentro de un escenario global de interacción entre distintas personas y máquinas. Distintos modelos de computación e interacción fueron estudiados para construir a su vez el modelo de máquinas sociales. Los formalismos finalmente escogidos y sobre los que se levanta el modelo formal de máquinas sociales son los de máquina de Turing, máquina de Turing persistente y π-cálculo. En la presentación del modelo se indica claramente cómo es que cada uno de estos formalismos se incorpora en el modelo de máquinas sociales. A continuación se presenta al lector la motivación y los objetivos con que este trabajo de título fue planteado desde un comienzo. Al final de este trabajo se evaluará el grado de cumplimiento de cada uno de estos objetivos.

Motivación. La motivación de este trabajo es la actual inexistencia de un sistema formal matemático que permita modelar las máquinas sociales, y la contribución tanto teórica como práctica que conllevaría su realización. Con una herramienta así, se realizaría una contribución importante al área de investigación, en términos de: describir con mayor rigurosidad los patrones de interacción y computación presentes en máquinas sociales; dar cuenta de las distintas clases de máquinas sociales en términos formales, asociándolas con respectivas propiedades en el modelo formal; y entregar un marco teórico que esté al servicio de los procesos de desarrollo de servicios y sistemas computacionales reales.

Objetivo general. El objetivo general de este trabajo es proponer una formalización matemática de la noción de máquinas sociales, basándose en modelos de computación e interacción existentes y que sean apropiados para lograr tal formalización.

Objetivos específicos. Los objetivos específicos de este trabajo son:

1. Estudiar las principales conceptualizaciones que hay en la literatura sobre la noción de máquinas sociales.

2. Revisar modelos de computación e interacción que se muestren pertinentes y adecuados para la formalización de máquinas sociales

3. Desarrollar un concepto formal matemático de máquina social, que se ubique correctamente dentro de los límites del concepto intuitivo de máquina social.

4. Evidenciar cómo el modelo desarrollado permite describir características prácticas de máquinas sociales reales a través de la expresión de propiedades formales que las modelen.

Finalmente se detalla dónde se desarrolla cada uno de estos objetivos específicos dentro de la estructura del presente documento. El primer objetivo específico se trabaja en los tres capítulos que componen la primera parte de la memoria. El segundo objetivo específico fue trabajado durante todo el período de tiempo que comprendió la elaboración del modelo formal de máquinas sociales, y se trabaja implícitamente en la segunda parte del documento. El tercer objetivo específico se trabaja extensamente y en detalle en la segunda parte del documento. Por último el cuarto objetivo específico se trabaja después de presentar el modelo de máquinas sociales, al final de la segunda parte del documento.

Parte I

Explorando la noción de máquinas sociales

En la primera parte de esta memoria se realiza una revisión del concepto de máquinas sociales. Esta noción es la pieza principal de todo nuestro estudio, por lo que resulta imperativo explorar su significado y el de sus ideas afines. Ese será nuestro primer objetivo. Nos permitirá posteriormente trabajar sobre un significado inicial, acotado y bien fundado de la noción de máquinas sociales. Ahora bien, hay otra razón por la que conviene explorar el significado de máquinas sociales. A diferencia de lo que pasa con otros objetos de estudio en ciencias de la computación, el concepto de máquinas sociales no tiene una definición establecida, acordada o transversalmente compartida entre los científicos. No se trata, por ejemplo, de términos ya tradicionales como «expresión regular» o «máquina de Turing». Tampoco se trata de un concepto ambiguo o inútilmente amplio. No. Sus principales enfoques muestran una gran similitud y concordancia. Pero aún así, la inexistencia de una definición tan precisa como transversal en la academia hacen aún más necesaria la revisión del concepto, ya que permite convenir también en los límites de su significado. En esta primera parte se explora la noción de máquinas sociales, se indaga su significado original, así como la de los principales enfoques que desde entonces se han ofrecido del concepto. No se intenta elaborar una revisión exhaustiva del tema, sino presentar las ideas fundamentales que gravitan a su entorno de modo que nos permitan comprender qué son las máquinas sociales y cuáles son las principales perspectivas desde las que éstas se investigan. Considerando los distintos matices conceptuales que habremos explorado, tendremos un significado inicial, acotado y bien fundando sobre la noción de máquinas sociales, que permitirá dar una base para el trabajo futuro. Para organizar mejor la exposición del tema, primero se revisará el contexto original desde donde nace el concepto de máquinas sociales, y se examinará su significado original. Luego se analizará brevemente el escenario general donde se sitúa la noción de máquinas sociales, y se revisarán los principales fenómenos que contextualizan el concepto que estudiamos y que han florecido a la luz de la nueva era tecnológica. Finalmente, se explorarán los principales enfoques con que distintos autores se aproximan al tema, lo que concluirá la construcción de esa anhelada noción inicial para comenzar nuestro trabajo en máquinas sociales.

Capítulo 1

El origen de las máquinas sociales

La noción de máquinas sociales nace como una propuesta prescriptiva sobre el rol de la Web en la sociedad. El término fue acuñado por Tim Berners-Lee en su famoso libro *«Weaving the Web»*, publicado el año 1999. En el contexto de una Web temprana, donde los esfuerzos se dirigían a impulsar más y más herramientas de colaboración entre las personas, el término *máquinas sociales* surge para proponer la creación de sistemas en la Web que fueran capaces de dirigir procesos de la sociedad misma, dejando a las personas la creación, y a sí mismos la administración. El escenario en que se gestó el concepto de máquinas sociales tuvo dos elementos constituyentes: un contexto histórico-tecnológico en el que se encontraba la Web, y un «sueño» del propio Berners-Lee sobre el rol de ésta en la sociedad. Primero se revisan estos elementos, luego se presenta el concepto de máquina social, y finalmente se analiza su significado original.

1.1. Contexto histórico-tecnológico

Desde sus primeros días de gestación en el CERN[1], la Web fue pensada como un espacio de información universal que permitiera a las personas colaborar y compartir información. Las personas serían capaces de leer y publicar cualquier pieza de información en este espacio, sin restricciones. La información estaría interconectada, y las personas podrían moverse de una pieza de información a otra con independencia de cuál fuera la tecnología subyacente. El soporte de toda pieza de información debía ser *equivalente*, y la Web abstraería la lectura y publicación de la información de su naturaleza tecnológica —e.g. tipo de computadores usados, sistemas operativos, sistemas de archivos, tipo de documentos, entre otros—. Para hacer esto posible se desplegó un gran y constante trabajo en el desarrollo de las ideas centrales de hipertexto y URI; en el protocolo de comunicación HTTP, primario aunque no exclusivo; y en una *lingua franca*, HTML, que asegurase el entendimiento entre los computadores. Estos pilares básicos dieron vida a la Web. Permitieron interoperar distintos tipos de documentos entre distintos computadores, permitieron a las personas obtener cualquier pieza de información existente en ese espacio —bajo la debida autorización—, y les permitiría también publicar

[1] `http://home.web.cern.ch/`

cualquier pieza de información, la que estaría inmediatamente disponible en el espacio para todos.

La Web pronto se abrió al mundo, y rápidamente comenzó a masificarse. Dentro de sus primeros años, esta Web temprana se enfrentaría a la dificultad de alinearse con su propósito inicial de contribuir a la colaboración y al compartir información. La existencia de los elementos técnicos mencionados arriba, y de la universalidad del espacio que estos permitieron crear, se debía esencialmente a una motivación social, y no tecnológica. Por eso, era claro que una Web distante de su norte colaborativo también estaba tecnológicamente incompleta. En la idea original de Berners-Lee, la Web requería ser un espacio en que fuera tan fácil acceder a la información como publicarla. El efecto social que se perseguía necesitaba que se pudiera leer y escribir con la misma sencillez. Sin embargo en su primera etapa la Web se vio enfrentada a un gran despliegue en la capacidad de acceder a la información, pero no así la de publicarla, y la colaboración por lo tanto estaba fuertemente amenazada. En 1999, Berners-Lee señalaba: «Con todo este trabajo en la presentación de contenido, aún nos hemos realmente ocupado sólo de la lectura de la información, no de la escritura de ésta. Hay poco para ayudar a utilizar la Web como lugar de encuentro colaborativo» (Berners-Lee & Fischetti, 2000, p. 169). El contexto tecnológico por el que pasaba la Web trajo consigo una situación en que una gran masa de usuarios se dedicaba solo a consumir la información de la Web, y unos poco se dedicaban a producirla. Como respuesta, el desarrollo de la Web estuvo entonces enfocado en corregir esta situación, y así se comenzó a trabajar en más y mejores herramientas que permitieran la colaboración entre las personas. En este contexto de un intenso despliegue en el desarrollo de herramientas de colaboración en la Web es que se gesta la noción que estudiamos.

1.2. Un sueño para la Web

Al momento de introducir la noción de máquinas sociales, Berners-Lee también había manifestado tener una clara visión sobre el rol que debía jugar la Web en la sociedad. Era una extensión social de las motivaciones iniciales que gestaron el desarrollo de la Web en el CERN. Esta visión es un sueño compuesto de dos partes. La primera parte consiste en una Web que se convierte continuamente en un mejor espacio para el trabajo grupal y la colaboración entre las personas. La segunda parte, que depende de la primera, se basa en los datos que se generan en la Web y que deben ser procesados como parte de las tareas cotidianas. Los computadores deben «ayudarnos a lidiar con el grueso de los datos, para hacerse cargo del tedio de todo lo que puede reducirse a un proceso racional, y para gestionar la escala de nuestros sistemas humanos» (Berners-Lee, 1997, párr. 8). Así, en la medida que la Web se convierte en un «espejo realista (o, de hecho, la encarnación más directa) de las formas en las que trabajamos y jugamos y socializamos», y en la medida en que nuestras interacciones se dan en línea —primera parte del sueño—, podemos «usar los computadores para ayudarnos a analizarlas, dar sentido a lo que estamos haciendo, dónde encajamos individualmente, y cómo podemos trabajar mejor juntos» (Berners-Lee, 1998, párr. 4) —segunda parte del sueño—.

En su libro *«Weaving the Web»*, Berners-Lee vuelve a manifestar su visión, explayándose con mayor ambición sobre las ideas de colaboración e interacción en la Web, y señalando a

la Web Semántica como el gran ente facilitador para alcanzar la segunda parte de su sueño:

> Tengo un sueño para la Web . . . y éste tiene dos partes.
>
> En la primera parte, la Web se vuelve un medio mucho más poderoso para la colaboración entre las personas. Siempre he imaginado el espacio de información como algo donde cualquiera tiene acceso inmediato e intuitivo, y no solo para navegar, sino también para crear. . . . El sueño de la comunicación persona-a-persona a través del conocimiento compartido debe ser posible para grupos de todos los tamaños, interactuando electrónicamente con tanta facilidad como lo hacen ahora en persona.
>
> En la segunda parte del sueño, la colaboración se extiende a los computadores. Las máquinas se vuelven capaces de analizar todos los datos de la Web. . . . Una "Web Semántica", que haría esto posible, tiene todavía que surgir, pero cuando lo haga, los mecanismos día a día del comercio, la burocracia y nuestra vida cotidiana serán manejados por máquinas hablándoles a máquinas, dejando que los humanos proporcionen la inspiración e intuición. (Berners-Lee & Fischetti, 2000, pp. 157–158)

Es importante hacer notar que el sueño de Berners-Lee y la Web Semántica son conceptos distintos, y se pueden presentar de forma completamente independiente. No obstante, y por la naturaleza de la visión de Berners-Lee, la Web Semántica encaja muy bien como medio para hacer realidad la segunda parte de su sueño. Como habilitador para que las máquinas se comuniquen y trabajen sobre significados humanos, la existencia de una Web Semántica permite a las máquinas tener una capacidad mucho más grande para ayudar en el análisis y proceso de los datos contenidos en los procesos sociales que se llevan en la Web. Pero hay que diferenciar cuidadosamente, reiteremos, entre los fines buscados y los medios propuestos.

1.3. Máquinas sociales

La noción de máquinas sociales surge como respuesta a la visión de Berners-Lee y el contexto histórico-tecnológico en que se encontraba la Web. Bajo un escenario donde se debía poner el esfuerzo en desarrollar más y mejores herramientas colaborativas, la posibilidad de hacer realidad la primera parte del sueño de Berners-Lee era trabajada. La segunda parte de su sueño, el uso de las máquinas para procesar los datos de las interacciones, debía también ser promovida, y así los esfuerzos debían considerar ambas partes de su visión. Entendiendo la potencial contribución social que esto podía traer, Berners-Lee introduce la noción de máquinas sociales como una manera de dirigir los esfuerzos en beneficio de las formas en que socializamos, y que implicaría trabajar en ambas partes de su visión. En una Web que necesitaba ser más colaborativa, el autor entendía cómo un nuevo espacio de interacción —que requería mejores herramientas—, y que podía favorecerse de máquinas que procesarán los datos generados, ayudaría a los procesos sociales. La noción de máquinas sociales nace entonces como una *proyección* al plano social del potencial uso de la Web que Berners-Lee soñaba. Berners-Lee introduce el concepto de máquinas sociales como sigue:

> Espero que estas herramientas se conviertan en un nuevo género común en la Web. La vida real está y debe estar llena de todo tipo de restricciones sociales —los procesos mismos desde los cuales la "sociedad" surge. Los computadores ayudan si los usamos para crear *máquinas sociales* abstractas en la Web: procesos en los que la gente hace el trabajo creativo y las máquinas hacen la administración. Muchos procesos sociales pueden ser mejor ejecutados por una máquina, porque la máquina está siempre disponible, está libre de prejuicio, y de cualquier forma, a nadie le gusta administrar ese tipo de sistemas. (Ib., p. 172)

Esta es la propuesta original de máquinas sociales. Además, esta propuesta va inmediatamente acompañada de algunos ejemplos de máquinas sociales y de usos potenciales. Entre estos ejemplos se encuentran sistemas de votación electrónica, sistemas para la revisión en línea por pares y juegos de rol multiusuario. Como ejemplo del potencial uso de estas máquinas, el autor señala la conducción de plebiscitos y la participación en la democracia. En una arista aún más interesante, Berners-Lee propone cómo una máquina social podría construirse a partir de otras, y cómo se podría generalizar el uso de una máquina «clonándola» para ser usada en una situación social equivalente. Esto lo explica usando el sistema del propio World Wide Web Consortium (W3C) como ejemplo:

> Podríamos construir la máquina social del consorcio usando las varias máquinas que forman grupos de trabajo y reuniones de personal y así sucesivamente. Podríamos admitir un conjunto de grupos de trabajo que puedan ser mostrados [al público] para formar un clúster auto-suficiente y ajustado para separarse y formar un nuevo par "clon" del consorcio. . . . En teoría, podríamos entonces generalizar esta nueva forma social. Entonces cualquiera podría empezar un consorcio, cuando las condiciones fueran correctas, al presionar unos pocos botones en la página Web de una "fábrica de consorcio" virtual. (Ib., p. 175)

1.4. Examinando la noción original

Corresponde ahora examinar la noción original de máquinas sociales introducida por Berners-Lee, y analizar su significado. Comenzamos dando un resumen sucinto de lo que será el análisis. La primera pregunta —y quizás la más básica— que tenemos que formular es: ¿*qué* son las máquinas sociales? La respuesta puede obtenerse textualmente del propio autor: «procesos en los que la gente hace el trabajo creativo y las máquinas hacen la administración». También podemos preguntar: ¿*quiénes* participan en estos procesos? La respuesta es así mismo inmediata: humanos y máquinas. A continuación preguntemos: ¿qué procesa ese proceso? ¿Cuál es su *finalidad*? Y nos encontraremos con los propios procesos sociales. Podríamos seguir con: ¿*dónde* encontramos las máquinas sociales? La respuesta nuevamente es evidencia de Berners-Lee: en la Web. Nuestra próxima pregunta podría ser: ¿cuál es la *naturaleza* de esos procesos Web? Aquí el autor también es claro, las máquinas sociales son procesos *abstractos*. Y luego, si preguntamos ¿quiénes son los *llamados a crear* las máquinas sociales? veríamos cómo es la propia computación como disciplina la llamada a desarrollar

todo esto dentro de un nuevo género. Analizamos esto a continuación. Antes una nota de advertencia: Berners-Lee utiliza cinco términos similares al introducir el concepto de máquinas sociales, a saber, *computadores*, *máquinas sociales* -a veces sólo máquina-, *máquina de cómputo* -al igual que el término anterior, a veces sólo máquina-, *proceso social* -a veces solo proceso-, y *proceso en la Web* -al igual que el término anterior, a veces solo proceso-. Se intentará ser lo más cauto posible al usar estos términos para no dar lugar a confusiones.

Qué son y quienes participan. La vida real está llena de distintos procesos sociales que existen con independencia de los medios y recursos disponibles para llevarlos a cabo. Entre esos medios se encuentran los procesos computacionales. Por su propia naturaleza, los computadores y sus procesos son en algunos casos preferibles como medio, por sobre el trabajo humano. Incluso la naturaleza misma donde se da el trabajo humano físico, como por ejemplo una oficina, puede tener más desventajas en algunos casos que, por ejemplo, la naturaleza virtual de la Web. Pero esto es solo una posibilidad, y todo depende del proceso social que se esté observando. Ahora bien, un proceso social —pensar en un proceso que el Servicio de Impuestos Internos debe llevar a cabo—, puede estar compuesto de muchas tareas, interconectadas o no, creativas o burocráticas, repetitivas por naturaleza o quizás únicas... *a priori* no se sabe. El caso es que parte de esas tareas pueden ser mejor realizadas por un computador, y otras mejor realizadas por las personas. Las máquinas sociales son procesos computacionales que permiten sacar el mejor potencial de las personas y los computadores. Estas máquinas son vistas como herramientas, recursos. Quienes participan en ellas son las propias personas y los computadores, y sus roles son claros: crear y administrar, respectivamente.

Qué hacen y con qué fin. En simple, una máquina social es un recurso, una herramienta Web, que permite llevar a cabo una parte o todo un proceso social. Aunque Berners-Lee no es explícito en mencionarlo, del contexto narrativo se deduce —y así lo interpretaremos— que las máquinas sociales son herramientas creadas con un propósito social en mente, donde por *propósito social* entenderemos cualquier forma de contribución en la socialización. Es decir, su finalidad es ayudar, auxiliar un proceso social. Ahora bien, la idea de usar sistemas en la Web para que las personas colaboren en línea y los datos generados sean administrados por las máquinas, no debe confundirse con el uso de herramientas Web para administrar datos como parte de un proceso social. Sino, pensemos en una calculadora o una aplicación de cálculo. Estas podrían ser fácilmente utilizadas como parte de un proceso en la sociedad, por ejemplo en el conteo de votos, la construcción de un hospitales, deducir la imposición de impuestos, etc. El hecho es que un ingeniero podría realizar sus tareas creativas de forma física —reuniones, conversaciones, toma de decisiones— y usar la máquina aisladamente. Incluso podría utilizar una herramienta en la Web para el cálculo en general. ¿Estamos en presencia de una máquina social? La respuesta es que no. Aquí es donde el término es distintivo de otras aplicaciones de la computación igualmente valiosas. La noción de máquinas sociales no es un enfoque de cómo los computadores facilitan los procesos sociales gracias sólo a sus capacidades de procesamiento —y no de colaboración—. No se trata de, extremando el argumento, utilizar un procesador de texto, imprimir una hoja, meterla a un sobre, correr cuadras para entregar un correo y volver corriendo a casa para seguir jugando una —dudosa— partida de rol en línea. No. La noción de máquina social es novedosa en el sentido que es una propuesta para desarrollar herramientas que realicen procesos sociales, o una parte de ellos. La máquina social es una herramienta para *reflejar fielmente* un proceso social, y en ese reflejo dejar el trabajo de cómputo a las máquinas y el trabajo creativo a las

personas. Este trabajo creativo debe darse en las máquinas sociales, no fuera. Esto debido a que la noción de Berners-Lee es prescriptiva: esta es la propuesta, el sistema debe integrar la parte colaborativa y administrativa de la tarea social. Si se deja fuera la colaboración o la administración simplemente no se está dentro de la propuesta, y por lo tanto no se está dentro de la noción original de máquinas sociales. Esto tiene también otra consecuencia: si las máquinas sociales están hechas para ayudar a los procesos sociales, es decir tienen un propósito, entonces los medios colaborativos y administrativos que ofrece una máquina social no pueden ser independientes o inconexos. Deben, en cambio, ser completamente conexos y coherentes. Si juntamos un conjunto de herramientas de colaboración con un conjunto de programas de cálculo y lo subimos a la Web difícilmente crearemos una máquina social. El significado original de la noción no lo abarcaría, simplemente porque no es lo que busca prescribir. Hay que notar, eso sí, que Berners-Lee no procede a definir proceso social, y por tanto podríamos entender el término en el sentido de cualquier forma de socialización.

Dónde habitan. Esta pregunta tiene una respuesta clara, que proviene del mismo Berners-Lee: habitan en la Web. Las máquinas sociales son definidas, en esta propuesta inaugural, como procesos en la Web. ¿Por qué esto es así? Si bien no hay una formulación explícita por parte del autor, su visión da algunas luces. Posiblemente la Web se sugiera como el espacio donde existan las máquinas sociales porque la Web es el espacio digital por excelencia que se propone para la colaboración. La noción de máquinas sociales requiere que las personas se puedan relacionar para así llevar a cabo un proceso social —además de que las máquinas administren y procesen—. La Web es esencial para garantizar la colaboración que implica el proceso social. Además, por la naturaleza de la Web, ésta puede disponer de la capacidad de cómputo de los computadores. La Web entonces aparece como una pieza fundamental en el significado original de las máquinas sociales, actuando como garante para que los procesos sociales puedan reflejarse satisfactoriamente.

Cuál es su naturaleza. Esta es quizás la pregunta que más deja a la interpretación. En lo que llevamos analizado hasta ahora, hemos estado rondando de alguna forma esta interrogante, aunque sin ser lo suficientemente taxativos en señalar a qué tipo de procesos en específico se refiere la noción de máquinas sociales. Responder esto es difícil ya que, lamentablemente, y como señalamos, Berners-Lee utiliza cinco términos similares al introducir la noción de máquina social. Por esto es mejor asumir y señalar que esta respuesta tiene una base en el autor, pero también una base en la interpretación propia. Así, para proceder con este análisis comenzaremos mencionando las ideas que son evidencia del propio autor, y luego las interpretaremos. Primero, las máquinas sociales son máquinas. Segundo, las máquinas sociales son procesos computacionales. Tercero, habitan en la Web. Cuarto y último, tienen el carácter de ser *abstractas* en ese espacio. Interpretemos lo anterior. Primero, y en general, el significado tradicional de máquina excluye en su naturaleza a quien la utiliza y/o para quien sirve. Un tren, por ejemplo, puede servir para conectar dos pueblos, llevar muchos pasajeros y ser manejado por un conductor, pero los pueblos, los pasajeros y el conductor no deben confundirse con el tren. Las máquinas son construidas por las personas, pueden contener personas, necesitar personas para funcionar de acuerdo a los planes originales, conectar personas, etc. Pero las máquinas, en su semántica común, no están *compuestas* de esas personas, así como un martillo sí está compuesto de un mango, o un auto de puertas. Por lo tanto, la naturaleza de los componentes de las máquinas sociales excluye a los seres humanos. Continuando, tenemos que las máquinas sociales son procesos computacionales. Estos procesos se crean usando

computadores. Más aún, estos procesos de origen computacional corren en la Web. Interpretando lo anterior, las máquinas sociales son procesos computacionales disponibles en la Web. Ahora, bajo la terminología clásica, un proceso computacional corresponde a un programa en ejecución. En consecuencia, estas máquinas se nos presentan, al menos hasta este punto, como programas en ejecución en la Web. Si seguimos incorporando lo que evidencia el autor, vemos que estas máquinas son de naturaleza *abstracta*. Este concepto es muy amplio, y lo interpretaremos como una prescripción sobre la naturaleza de estas máquinas, de *abstraerse* de su implementación y participantes, y definirse por el proceso social que se desea realizar y por sus propiedades. Esto quiere decir que estos programas en ejecución en la Web deben ocultar su implementación y trabajar como unidad de acuerdo a la finalidad que persiguen. Así, las personas deben ser capaces de participar en estas máquinas sociales sin restricciones de implementación, solo con las limitaciones propias de la socialización que se refleja. Además, estos procesos computacionales en la Web se corren en computadores conectados entre sí por diversos protocolos. Y de esa forma, esos procesos pueden utilizar la capacidad de cómputo de los computadores que los ejecutan —o de cualquier otra máquina a la que puedan acceder—, para así administrar y procesar los datos colaborativos que las personas generan. De esta forma, las máquinas sociales son verdaderos sistemas Web que permiten a las personas involucradas en un proceso social colaborar, abstrayéndose de los programas en ejecución que conforman tales sistemas, los que a su vez utilizan el poder de cómputo de las máquinas sobre las que corren para administrar y procesar los datos necesarios.

Un llamado a la computación. ¿Quiénes son los llamados a trabajar en las máquinas sociales? Berners-Lee hace un llamado, una propuesta sobre cómo los computadores ayudarían si creamos máquinas sociales. Esa propuesta, es una propuesta para la gente que trabaja en computación, tanto en la práctica como en la teoría, tanto en la industria como en la academia. «Espero que estas herramientas se conviertan en un nuevo género común en la Web». Berners-Lee es bastante claro: está haciendo un llamado a la disciplina de la computación a desarrollar un nuevo tipo de herramientas. Éstas son herramientas de colaboración, que deben ser pensadas como un nuevo género en el desarrollo Web, y donde se van a encontrar las máquinas sociales. Hoy podemos decir que aunque este llamado tardó en ser escuchado, permitió que este concepto comenzara a definir un área de estudio, donde las máquinas sociales se volvieron un objeto de estudio y un tópico propio en la computación.

No podemos terminar este análisis sin antes mencionar un punto que se nos queda en el tintero. Esto tiene que ver con el carácter unitario y plegable de las máquinas sociales. Las máquinas sociales son pensadas por Berners-Lee como estructuras autocontenidas, y en ese sentido, pueden verse como piezas disponibles para formar otra pieza mayor que las contenga. Berners-Lee propone estas máquinas como unidades que se pueden componer, intercambiar y clonar. Este concepto de *pieza de lego clonable* es mencionado como ejemplo y de forma algo ligera por parte del autor. Sin embargo, hoy en día es uno de los principales enfoques en el estudio de máquinas sociales. El cómo componer máquinas sociales, y cómo componer así procesos sociales, es algo a lo que volveremos al finalizar esta primera parte, y veremos el gran énfasis que esto ha alcanzado en el estudio de las máquinas sociales.

Capítulo 2

Revisión de fenómenos emergentes en la Web que proveen contexto a la noción de máquinas sociales

La propuesta original de Berners-Lee de la noción de máquinas sociales tuvo lugar en un tiempo donde la Web otorgaba herramientas de colaboración e interacción por debajo a las que hoy en día conocemos. Debido a los continuos esfuerzos en el desarrollo Web, poco a poco fueron apareciendo nuevas herramientas de colaboración, dejando en el camino la situación inicial de una gran cantidad de personas que consumía lo que pocos alcanzaban a publicar. Esto permitió que más procesos sociales pudieran darse en línea. Pero esta evolución también trajo consigo la creación de nuevas formas de socialización *genuinamente* Web, que antes no existían. Así como la invención del teléfono alteró el concepto de comunicación, la evolución de la Web alteró nuestro concepto de socialización. Estos nuevos fenómenos que emergieron producto del impacto social y tecnológico de la Web comenzaron a ser estudiados por distintas disciplinas académicas y bajo el amparo de distintas áreas de estudio. Así, el escenario donde se encuentra la noción de máquinas sociales se configura como un espacio de estudio donde cohabitan otros conceptos que dan cuenta desde distintas perspectivas del impacto social y tecnológico que ha tenido la Web. Dicho de otra forma, el concepto de máquinas sociales hoy en día es uno más dentro de un conjunto de conceptos similares que estudian aspectos específicos relacionados a las formas de socialización que la Web ha generado. Así, para comprender el escenario general donde se ubica la noción de máquinas sociales se debe ampliar la mirada y revisar, aunque sea brevemente, los principales fenómenos y conceptos que forman el contexto donde habita actualmente la noción que estudiamos. Hacemos esto en lo que sigue.

2.1. Prosumo en la Web

La evolución que tuvo la Web generó un profundo cambio en el modelo de producción y consumo de contenido que las personas mantenían anteriormente. Un nuevo software social

se caracterizó por la presencia del contenido generado por usuario, tanto por posibilitarlo, como por necesitarlo. La relación de las personas con la Web cambió; lo que llega a conocerse como Web 2.0 tiene dentro de sus fundamentos el cambio productor-consumidor que trajo consigo la evolución del software social. Este cambio corresponde al establecimiento de una relación simétrica y equilibrada entre las actividades consumo y producción de contenido por parte de los usuarios, y corresponde al primer fenómeno que se revisará; además representa una pieza clave para comprender otros fenómenos que han surgido, y que no eran plausibles bajo la relación que se tenía en la Web 1.0. Ahora bien, mixturas entre producción y consumo similares a ésta han ocurrido con anterioridad en otras esferas del quehacer humano. La igualdad de condiciones en el rol de productor y consumidor ha sido objeto de estudio por bastante tiempo. Paradójicamente quizás, el estudio de este fenómeno en los términos tradicionales de producción y consumo ha mostrado ser insuficiente, y un nuevo enfoque que asume una continuidad entre estos polos ha surgido. Así, el fenómeno que nos ocupa ha sido largamente estudiado bajo esta nueva perspectiva, que responde al concepto de *prosumerism*. Lo estudiamos a continuación.

Con origen en la economía y las ciencias sociales, los conceptos de *prosumer* y *prosumption*[1], nacen para referirse a quien es a la vez productor y consumidor, y al proceso subyacente donde se combina la producción y el consumo, respectivamente. La terminología fue acuñada por Alvin Toffler en 1980. Ritzer, Dean y Jurgenson (2012) explican:

> Los humanos son por su propia naturaleza prosumers (e.g., aquellos en sociedades de caza y recolección son mejor pensados como prosumers), y la existencia de consumidores y productores en gran parte separables es, en el mejor de los casos, una anomalía histórica. Esto es, puede ser que hubiera buenas razones para afirmar que los trabajadores de la fábrica en el apogeo de la Revolución Industrial pudieran ser pensados como productores, y los compradores en los Estados Unidos en los 1970s como consumidores, pero tales pensamientos están incrustado en, y limitados a, circunstancias históricas específicas. (p. 380)

La producción y el consumo se dan de forma conjunta, en oposición a una visión binaria y excluyente. Los autores grafican el prosumo como un «contínuo producción-consumo» donde en la mitad de los dos extremos —producción y consumo— está el «prosumo puro». Ritzer et al. analizan la práctica del prosumo —el posicionamiento en el centro de ese continuo— centrándose en la enorme expansión que ha tenido en los últimos años como respuesta a una serie de cambios sociales fundamentales. Estos cambios les permiten entender con mayor claridad la naturaleza del prosumo. A continuación revisaremos algunos de ellos desde la perspectiva de estos autores. [*Fuente de la siguiente revisión*: (Ritzer, Dean & Jurgenson, 2012, pp. 381–386)]

Producción inmaterial. Uno de los cambios sociales más importantes que analizan Ritzer et al., es el vuelco de una producción fundamentalmente material, a una cada vez más inmaterial. Lo ejemplifican con el caso de la industria automotriz, donde la producción material de autos es, según los autores, de menor importancia que «las ideas para mejorar la manufactura, marketing, o diseño de los productos» (Ib., p. 382). Una producción que requiere poco o

[1] En adelante, prosumer y prosumo.

ningún labor material puede salir de las «murallas de la fábrica» y entrar en «la sociedad como un todo, creando la "fábrica social" o una "fábrica sin murallas"» (*Loc. cit.*). Como esta producción inmaterial se da en el reino de las ideas y ya no encerrada en la fábrica, se abre la posibilidad para que los consumidores de esas ideas puedan también producirlas. Ritzer et al. indican como ejemplos de lo anterior el movimiento open source, citando a Linux y Firefox; también a los métodos de publicidad que solicitan a los consumidores que les proporcionen ideas; e igualmente a la producción del significado compartido que tienen las marcas. En este último caso los prosumers producen —a la vez que consumen— el significado que acompaña a las marcas como McDonald's, BMW y Nike: éste no puede ser producido por la gente de marketing e impuesto a las personas, sino que debe ser creado por los mismos prosumers. Los autores agregan: «quizás las últimas fábricas sociales son los sitios de la Web 2.0 donde los prosumers simultáneamente consumen y producen ideas en, por ejemplo, wikis, blogs . . . y sitios de redes sociales» (Ib., p. 383).

Clientes como empleados. Otro cambio que ha fomentado la práctica del prosumo y, que en términos de los autores, facilita ver las limitaciones de la distinción entre producción y consumo, es el aumento de servicios que «ponen al cliente a trabajar». El ejemplo clásico son los clientes de locales de comida rápida como McDonald's, donde se los fuerza a hacer filas para obtener la comida y preocuparse de sus bandejas al momento de salir. De esta forma los clientes hacen trabajo realizado en el pasado por empleados del local —como mozos en restaurantes—, permitiendo la eliminación de esos empleados, y convirtiendo a los consumidores en productores. El mismo hecho se representa, con las diferencias pertinentes, en locales de cena buffet, como Gatsby, cafeterías como Starbucks, y en los supermercados modernos. Este último caso representa una interesante visión sobre la evolución del pasivo consumidor de almacén que solicita al vendedor activo los productos, al prosumo que se da en los supermercados, donde el consumidor realiza el trabajo productivo de recolectar, empacar e incluso en algunos casos pasar por cajas de autoservicio.

Desarrollo tecnológico. Muy relacionado con lo anterior está el rol del cambio tecnológico, que ha dado mayor importancia al prosumer. Muchas de las distintas tareas que los trabajadores antes realizaban por los consumidores pueden ser hoy llevadas a cabo por parte de los prosumers través de la tecnología. Los trabajadores que quedan están destinados principalmente a dar soporte y ayudar a los prosumer con la nueva tecnología. Aquí el ícono son las cajas de autoservicio y, en especial, los cajeros automáticos (ATM).

Economía de la experiencia. El giro a una economía basada en la experiencia es otro cambio promotor del prosumo. Aquí «los consumidores juegan un rol crucial y activo en la producción de sus propias experiencias» (Ib., p. 384). Los autores usan como ejemplo unas vacaciones en Disney World, donde «estamos simultáneamente consumiendo las experiencias que los "imagineers" de Disney quieren que tengamos *y* produciendo nuestras propias experiencias únicas, a partir de la realidad que tenemos ante nosotros, así como de nuestras únicas historias y realidades presentes» (Ib., pp. 384–385).

Impacto del Internet y la computación. El último cambio que social que revisaremos es el surgir del Internet[2] y el software social. Los autores examinan el fenómeno específico con que iniciamos esta revisión: el prosumo en la Web. Desde su punto de vista el cambio social

[2]En este punto los autores usan las nociones de Internet y Web de forma intercambiable.

introducido por la tecnología Web es primordial. Ellos señalan: «un cambio tecnológico absolutamente crucial que necesita ser subrayado separadamente es la creciente importancia de los computadores y el Internet para los procesos de prosumo» (Ib., p. 385). Los autores analizan primero la naturaleza del trabajo en Internet, señalando que los sitios, al no contar con trabajadores presentes inmediatamente, y al ser difícil o imposible ubicarlos telefónicamente, delegan en los prosumers la responsabilidad de hallar los productos, evaluarlos, ordenarlos y pagarlos. Esa situación es un cultivo perfecto para el prosumo: «En tanto más y más consumo tradicional se vuelca a Internet, donde es difícil o imposible encontrar trabajadores tradicionales, es cada vez más claro que el prosumo es lo que define mucho del Internet» (*Loc. cit.*). Inmediatamente después, los autores tratan la producción de contenido en la Web 2.0:

> El prosumo en Internet ha ocurrido cada vez más a través del contenido generado por usuario en lo que se ha llegado a conocer como Web 2.0 (en la Web 1.0, tal como AOL o Yahoo, el contenido es generado por el productor, dejando un pequeño espacio para el prosumo). La Web 2.0 incluye la Web social con sitios como Facebook y Twitter, la blogósfera, Wikipedia, sitios para compartir contenido como Flickr y YouTube, y muchos más donde los usuarios no solo consumen sino también producen contenido. Es en los mundos inmateriales de la Web 2.0 donde es más difícil distinguir entre productores y consumidores, donde la hegemonía de los prosumers es más clara. (*Loc. cit.*)

[*Finaliza la revisión*]

2.2. Wisdom of crowds

La evolución de la Web ha estado marcada notablemente por la aparición creciente y sustantiva de fenómenos que revelan con más claridad el potencial intelectual que manifiestan diversos grupos que socializan en la Web. En lo que sigue revisamos dos conceptos bajo los cuales se estudia el potencial intelectual colectivo que puede ser elicitado a través de la Web. Estos conceptos se fundamentan en un marco de estudio mucho más amplio sobre el potencial intelectual de comunidades, masas y grupos en general, y no se reducen sólo a los fenómenos que exhibe la Web. Aún así hoy en día forman parte fundamental del contexto de máquinas sociales. Estos conceptos corresponden al de *wisdom of crowds* e inteligencia colectiva. A continuación revisamos el primero de ellos. [*Fuente de la siguiente revisión*: (Surowiecki, 2005, pp. xiii–142)]

El término *wisdom of crowds* fue acuñado por James Surowiecki en su libro *«The Wisdom of Crowds»*, para referirse a la capacidad que tienen las multitudes de personas para desempeñarse, bajo las condiciones adecuadas, con mayor inteligencia que hasta la persona más inteligente del mismo grupo. Históricamente, señala Surowiecki, siempre se ha considerado que las personas son inteligentes y las multitudes estúpidas. Como el autor evidencia, esta idea ha gravitado en disciplinas como la filosofía, la historia, la política, el periodismo y las letras. Pero esta visión ha imposibilitado reconocer la inteligencia y efectividad para resolver problemas, cooperar y tomar decisiones que las multitudes verdaderamente presentan. Surowiecki realiza un compendio analítico de estudios de diferentes disciplinas —psicología, sociología, estadística, simulación de agentes por computador, economía experimental, en-

tre otros— que devela el enorme potencial intelectual de los grupos o multitudes dadas las condiciones adecuadas[3]. El autor explica:

> Bajo las circunstancias correctas, los grupos son extraordinariamente inteligentes, y son generalmente más inteligentes que la persona más inteligente en ellos. Los grupos no necesitan ser dominados por personas excepcionalmente inteligentes a fin de ser inteligentes. Incluso si la mayoría de las personas en un grupo no son especialmente bien-informadas o racionales, éste [grupo] todavía puede alcanzar una decisión colectivamente sabia. (Ib., pp. xiii-xiv)

Una multitud puede enfrentarse, según el autor, a tres clases de problemas: los problemas de cognición, donde las soluciones son definitivas y/o las respuestas pueden ser mejores o peores que otras —e.g. «¿Quién ganará el *Super Bowl* este año?», «¿Cuál es el mejor lugar para construir esta nueva piscina pública?»—; los problemas de coordinación, donde los miembros de la multitud deben encontrar una forma de coordinar sus comportamientos con el resto —e.g. «¿Cómo los compradores y vendedores se encuentran y comercian a un precio justo?», ¿Cómo conducir y evitar el tráfico?—; y finalmente, los problemas de cooperación, que giran en torno a la tensión que se produce entre las fuerzas del interés personal versus el interés grupal —e.g. pagar impuestos, utilizar leña en una ciudad, entre otros—. Sin embargo, para que la multitud sea sabia, señala Surowiecki, deben darse las siguientes condiciones: *diversidad*, *independencia*, y *descentralización*. En lo que resta analizamos estos tres tipos de problemas, y las tres condiciones necesarias para que emerja la sabiduría de la multitud.

Problemas de cognición. En este tipo de problemas, y bajo las condiciones mencionadas, la multitud esconde un enorme potencial para agregar su información y obtener soluciones más exactas que cualquiera de sus integrantes. Los primeros estudios vienen de la sociología y sicología en el apogeo de la investigación en dinámica de grupos. Aquí los experimentos reunían grupos de personas y les solicitan cuantificar variables: adivinar la temperatura actual, ordenar ítemes por peso, etc. Los resultados reflejaban que si se agregaban y promediaban las respuestas, ese valor se encontraba generalmente dentro de los más cercanos a la realidad, entre los otros valores de los integrantes del grupo. Esto rápidamente empezó a ser extensamente estudiado, y abarcó todo un período desde 1920 a 1950. El autor resalta dos cosas: primero, las respuestas que se agregan son elaboradas de forma individual, sin conversación ni diálogo de por medio. Segundo, la «adivinanza grupal» no era siempre la mejor, pero ningún individuo, tampoco, superaba al resto del grupo en cada momento. Estos experimentos y este fenómeno representa un punto de apoyo esencial para el autor y la idea de *wisdom of crowds*, sobre todo en los problemas de cognición:

> En otras palabras, si tu corres diez experimentos diferentes de *jelly-bean-counting* [conteo de dulces en frascos], es probable que cada vez uno o dos estudiantes superarán al grupo. Pero ellos no serán los mismos estudiantes cada vez. Sobre los diez experimentos, el desempeño del grupo será casi seguro el mejor del grupo. La forma más simple de obtener confiablemente buenas respuestas es tan solo preguntar al grupo cada vez. (Ib., p. 5)

[3]El autor usa los términos grupo y multitud —masa también— de forma intercambiable, y el significado que adoptan es bastante amplio, variando en el rango de audiencias de TV, corporaciones, apostadores de carreras, equipos de administración, personas atrapadas en un tráfico, entre otros.

Las bondades de agregar las respuestas individuales también han sido estudiadas mediante simulación computacional, como presenta Surowiecki al referirse a los experimentos de Norman L. Johnson en Los Alamos. Aquí se simulan agentes en laberintos y luego se compara con la solución agregada. Los resultados llevan a la misma conclusión. También lo hacen, según el autor, las apuestas de deportes. Desde el punto de vista de predecir resultados futuros, el mercado de las apuestas es un excelente ejemplo. «Alrededor de tres cuartas partes de las veces, la última línea del Mirage [centro de apuestas en las Vegas] será el pronóstico más fiable de los resultados de los juegos de la NFL que se pueda encontrar» (Ib., p. 15). De hecho, el mercado de las apuestas ha mostrado ser tan eficaz en predecir el futuro que ha recibido gran atención académica, donde se lo usa para predecir con mayor precisión interrogantes de distintas índoles. Por ejemplo, el Iowa Electronic Markets (IEM) de la Universidad de Iowa, consistentemente ha utilizado mercados de apuestas para predecir en EE. UU. los resultados de elecciones presidenciales, de congresistas y de gobernadores, entre otros. El desempeño de estos mercados, indica el autor, es sorprendente: entre 1988 y 2000 los precios de la víspera de las elecciones en el IEM erraban por solo 1,37 % en elecciones presidenciales, 3,43 % en otras elecciones en EE. UU., y 2,12 % en elecciones extranjeras. Incluso, en tres cuartas partes de las veces, el mercado de apuestas supera al de las encuestas.

Uno de los ejemplos más importantes que Surowiecki da respecto al *wisdom of crowds* es el de la búsqueda en la Web. Este problema es claramente cognitivo, y la forma en que día a día lo resolvemos es usando el *wisdom of crowds*. Para generar el resultado de una búsqueda en la Web, Google realiza una agregación como las mencionadas al inicio de este apartado, dejando en evidencia, día tras día, la sabiduría que hay en las multitudes. El algoritmo base usado por Google, PageRank, consiste —muy simplificadamente y bajo la perspectiva de *wisdom of crowds*— en interpretar un link de una página A a una B como un «voto» de A a B. Según los votos que tenga una página, se establece su «importancia». En la medida que una página es más importante, su voto «pesa» más. Al final, la importancia de una página se establece por sus votos ponderados por el peso de estos. Surowiecki señala:

> [En una búsqueda,] lo que Google está haciendo es pidiendo a la Web entera que decida cuál página contiene la información más útil, y la página que obtiene más votos va primera en la lista. Y esa página, o la inmediatamente debajo de ella, con mucha frecuencia es de hecho aquella con la información más útil.
>
> Ahora, Google es una república, no una democracia. Como dice la descripción [de PageRank], mientras más personas han enlazado una página, mayor influencia tiene esa página en la decisión final. El voto final es un "promedio ponderado". . . . No obstante, los sitios grandes que tienen más influencia sobre el veredicto final de la multitud tienen esa influencia sólo debido a todos los votos que los sitios más pequeños les han dado. Si los sitios más pequeños estuvieran dando mucha influencia a los sitios equivocados, los resultados de búsqueda de Google no serían exactos. Al final, la multitud todavía gobierna. Para ser inteligente en la cima, el sistema tiene que ser inteligente a través de todo el camino. (Ib., pp. 16–17)

Problemas de coordinación. Este tipo de problemas se identifican porque para resolverlos, cada persona tiene que pensar no sólo en cuál es la respuesta correcta, sino también en cuál es la respuesta que los otros creen correcta. El autor da como ejemplos el *problema pedestre*

—cómo podemos caminar por una vereda en la que transitan otras personas, adelantando y parando sin transformar todo en caos—, la elección de la hora en que las personas salen trabajar, cómo se asignan los asientos en el metro, entre otros. En el corazón de estos problemas, señala Surowiecki, está la pregunta: «¿Cómo puede la gente voluntariamente —esto es, sin nadie diciéndoles qué hacer— lograr que sus acciones encajen entre sí en una forma eficiente y ordenada?» (Ib., p. 86).

En base a una revisión de estudios científicos, el autor va reconociendo elementos que inciden en la coordinación grupal. Entre estos se encuentran los *puntos de Schelling*, que son puntos de referencia comunes a personas de una misma cultura y permiten a las personas coordinarse sin recibir órdenes e incluso sin hablarse entre sí —e.g. puntos de encuentro en un pueblo, horarios en los que se llevan a cabo tales o cuales acciones, entre otros—. También se encuentran las convenciones sociales, que «reducen la carga de trabajo cognitivo» y permiten a personas desconectadas «organizarse entre sí con relativa facilidad y en ausencia de conflicto» (Ib., p. 93).

El potencial de la coordinación es alto, señala el autor. Aún con una capacidad intelectual limitada, la coordinación que exhiben los estorninos es un ejemplo de ello. Ninguno comanda a otro, y a través del movimiento individual que cada uno realiza, permiten a la bandada como conjunto migrar en la dirección correcta, esquivar a los depredadores y reagruparse cuando se dividen. De la misma forma, el autor plantea un problema importante de coordinación, a saber, el ofrecer los bienes y servicios en la sociedad en el lugar, la hora y con el costo correcto. En este sentido el autor analiza por qué es posible encontrar jugo de naranja en un supermercado, que fue producido, embotellado y empacado días antes. La pregunta es cómo es posible que el productor supiera de antemano quién consumiría, cuándo y a qué precio. Para el autor este es posiblemente el problema de coordinación más importante, y la forma de resolverlo es mediante el mercado. Los integrantes de la cadena usan su conocimiento local, y actúan individualmente, como los estorninos, logrando coordinar las actividades económicas.

Problemas de cooperación. Los problemas de cooperación, señala Surowiecki, pueden parecerse a los de coordinación, ya que en ambos casos las personas deben prestar atención a lo que otros hacen. Pero, a diferencia de los problemas de cooperación, los problemas de coordinación podían resolverse aún cuando cada persona buscara el interés propio. No es así en el caso los problemas de cooperación. Surowiecki explica:

> Para resolver problemas de cooperación —los que incluyen cosas como mantener la vereda libre de nieve, pagar impuestos, y reducir la contaminación— los miembros de un grupo o una sociedad necesitan hacer más. Necesitan adoptar una definición más amplia de interés propio que aquella miope de maximizar ganancias en las demandas a corto plazo. Y necesitan ser capaces de confiar en aquellos que los rodean, porque en ausencia de confianza el perseguir el miope interés propio es la única estrategia que tiene sentido. (Ib., pp. 110–111)

Hay diversos elementos que inciden en los problemas de cooperación, según muestra el autor. Entre estos está la llamada *reciprocidad fuerte*, que es «la disposición a castigar el mal comportamiento (y premiar el buen comportamiento) aún cuando no se obtienen beneficios materiales personales al hacerlo» (Ib., p. 116). Este fenómeno es un ejemplo de *comportamien-*

to prosocial, ya que «empuja a las personas a trascender una definición estrecha de interés propio y hacer cosas, intencionadamente o no, que terminan sirviendo al bien común» (*Loc. cit.*). Otro hecho que destaca el autor cómo fundamental es el aprendizaje histórico de la sinergia de la cooperación, en donde todos terminan siendo beneficiados. Para el autor, esto es un hecho sumamente significativo ya que es lo que nos permite cooperar con desconocidos. Surowiecki destaca:

> Lo interesante es que cooperamos con desconocidos. Hacemos donaciones de caridad. Compramos cosas en eBay sin haberlas visto. La gente se registran en Kazaa y suben canciones para que otros las descarguen, aun cuando ellos no obtienen ningún beneficio por compartir esas canciones y hacerlo significa permitir que extraños tengan acceso al disco duro de sus computadores. Estas son todas, en el sentido estricto, cosas irracionales de hacer. Pero ellas nos hacen a todos (bueno, fuera de las compañías discográficas) mejores. (Ib., p. 118)

Finalmente, otro elemento a considerar es la *reciprocidad* —a secas— o *consentimiento contingente*. Esto, explica el autor, nos protege del incentivo a sentarnos y esperar que otros hagan las cosas por uno, es decir, a no cooperar y aprovecharnos de los beneficios del grupo. Es lo que nos permite, por ejemplo, admitir conductas de cooperación como el pago de impuestos. Surowiecki describe cómo el sistema de impuestos de EE. UU. *necesita* personas que quieran pagar sus impuestos, dado las escasas chances de ser atrapado por evasión. Aquí las personas pagan impuestos en la medida que ven que los otros también lo hacen, y que aquellos que no, son castigados. La confianza en el sistema tiene un profundo impacto en cómo se comportan estos consentidores condicionales. Atrapar realmente a los evasores puede resultar tan importante como la imagen pública del sistema. El autor revela estudios de economía experimental que permiten ubicar a las personas en una de tres categorías: egoístas, altruistas o consentidores condicionales. Los egoístas tienden a actuar según el interés propio en su sentido más estrecho, los altruistas cooperarán aún cuando otros no lo hagan, mientras que la mayor parte serán consentidores condicionales, quienes cooperarán en la medida que otros cooperen y los que no, sean castigados. De ellos depende mayoritariamente el éxito de la cooperación grupal.

Diversidad. La primera condición que Surowiecki señala como necesaria para el *wisdom of crowds* es la diversidad. La diversidad —en un sentido cognitivo— es fundamental para el buen desempeño de los grupos y las multitudes. La diversidad es la principal fuerza que permite enfrentar problemas donde las soluciones posibles se desconocen. Es la herramienta, como muestra Surowiecki, que hace de una colmena de abejas un colectivo brillante, y es la fuerza que permite a la humanidad abrirse espacio entre la tecnología aún no descubierta, seleccionando la más adecuada. También es lo que nos permite superar los sesgos que aparecen en grupos poco heterogéneos y en cambio aprovechar las bondades de cada individuo. Revisemos brevemente un caso de estudio que propone el autor, y que aplica posteriormente a la noción de *wisdom of crowds*.

A veces llamada inteligencia de colmena, el comportamiento inteligente y colectivo que las abejas exhiben para recolectar néctar les permite realizar búsquedas en áreas con radios de hasta de 6 kilómetros de largo a partir de la colmena, teniendo más de un 50 % de probabilidades de encontrarlo en el área dentro de los primeros 2 kilómetros de radio. Para

lograrlo, las diminutas abejas envían exploradoras al entorno, y cuando éstas encuentran néctar vuelven al panal en busca de «refuerzos». Lo que sucede entonces es que éstas danzan para atraer la atención de otras abejas, siendo su danza tan intensa como la calidad del néctar. El resultado es que un nuevo grupo de abejas acompaña a explorar la zona donde se encontró el néctar y así, las abejas terminan explorando más intensamente las áreas con más néctar.

El problema que enfrentan las abejas no es muy distinto del que enfrentan los grupos en muchos casos. Se distinguen porque no se puede comenzar analizando todas las alternativas para determinar un plan ideal. Y no se puede, señala Surowiecki, porque no se tiene ninguna idea sobre cuáles son las posibles alternativas. El autor los describe como procesos de dos etapas. Primero, descubrir las alternativas posibles. Segundo, decidir entre ellas. Los avances tecnológicos como el automóvil, el ferrocarril o los computadores personales, apunta Surowiecki, son un buen ejemplo: los mercados inicialmente «están caracterizados por una abundancia de alternativas, muchas de ellas dramáticamente diferentes entre sí en diseño y tecnología. Con el paso del tiempo, el mercado identifica los ganadores y los perdedores, efectivamente escogiendo cuáles tecnologías florecerán y cuáles desaparecerán» (Ib., p. 26). Lo que en las abejas sucede al descubrir nuevas fuentes de polen, aquí sucede al descubrir nuevas tecnologías —e.g. automóvil en base a gasolina, a electricidad y a vapor—. Luego, la multitud da su veredicto. En el caso de las abejas, en base a la intensidad de su baile, en el caso tecnológico, en base al mercado —e.g. predominio del automóvil en base a gasolina—, como indica Surowiecki.

Estos ejemplos exhiben la importancia de la diversidad en la multitud. Aquí la diversidad juega un doble rol: primero, permite a los individuos encontrar soluciones diversas; segundo, permite a la multitud retroalimentar positivamente las mejores soluciones y explorarlas con mayor amplitud. En ambos casos, a mayor diversidad, mejor desempeño del grupo. El autor señala:

> Uno desea diversidad entre los emprendedores que están apareciendo con nuevas ideas, para que así se termine con diferencias significativas entre aquellas ideas, en vez de variaciones menores sobre el mismo concepto. Pero uno también deseas diversidad entre las personas que tienen el dinero. . . . Mientras más similares sean, más similares serán las ideas que ellos aprecien, y así el conjunto de nuevos productos y conceptos que el resto de nosotros verá será más pequeño que el posible. En contraste, si ellos son diversos, las chances de que al menos alguno apostará por una idea radical o poco probable obviamente aumenta. (Ib., p. 28)

Pero la diversidad no sólo sirve para añadir perspectivas que de otro modo no estarían presente: también sirven para evitar lo que Surowiecki denomina «características destructivas de la toma de decisiones grupales». Aquí el autor realiza una amplia revisión de estudios psicológicos y sociológicos, con el fin de establecer las principales limitaciones que la falta de diversidad genera en los grupos. Entre estas el autor menciona el sesgo y la influencia indebida que pueden ejercer algunos individuos sobre la decisión colectiva; la conformación de grupos que seleccionan a «los más inteligentes», quienes al parecerse entre sí se desempeñan peor como grupo que los que reúnen personas con menos habilidades pero más diversas; la homogeneidad grupal, que favorece lo que todos saben hacer bien pero desfavorece la in-

vestigación de alternativas. En este último caso, señala el autor, incorporar a un individuo distinto, aún cuando sea inexperto y menos capaz, hace al grupo más inteligente sólo porque no es redudante con los otros. Además la homogeneidad fomenta la conformidad, un fenómeno bastante estudiado donde la presión que existe en los individuos para sentirse parte del grupo los lleva a omitir su opinión o incluso cambiarla. Otra de estas características destructivas es el *groupthink*, donde el grupo es tan homogéneo y coheso, que los individuos se convencen de que los juicios que el grupo genera son los correctos. El grupo se aísla y, o bien excluye o bien racionaliza como un error toda posible información que represente un desafío, reforzando su dependencia en el grupo y convenciéndose más de estar en lo correcto. Aquí se destruye cualquier información divergente. No obstante, Surowiecki subraya que no es que no existan los expertos. Estos sí existen, lo que sucede es que su expertiz es extremadamente reducida a un propósito intelectual específico, por lo que el desempeño de los expertos «variará dramáticamente dependiendo del problema que se les pida resolver» (Ib., pp. 34–35). Además tienden a sobrestimar las posibilidades de estar en lo cierto: los estudios muestran que los expertos —en distintas áreas— creen saber más de lo que saben. También muestran que por sobre cierto nivel de conocimiento, la expertiz y la precisión no están relacionadas.

Independencia. La segunda condición que Surowiecki señala como necesaria para el *wisdom of crowds* es la independencia de opinión entre los integrantes de la multitud, en el sentido de librarse de la posible influencia que otros puedan ejercer. Cada persona tiene «información privada», producto de sus propias interpretaciones, análisis, intuiciones, etc., y es esa la información que combinada permite a la multitud ser inteligente. Mientras mayor sea la influencia que en una multitud los unos ejerzan sobre los otros, más posibilidades hay de creer las mismas cosas y cometer los mismos errores, señala el autor. Surowiecki explica:

> La independencia es importante para la toma de decisiones inteligentes por dos razones. Primero, evita que los errores de las personas lleguen a estar correlacionados. Los errores en el juicio individual no arruinarán el juicio colectivo del grupo en tanto esos errores no estén sistemáticamente apuntando en la misma dirección. Una de las formas más rápidas de hacer los juicios de las personas sistemáticamente sesgados es hacerlas dependientes entre sí para obtener información. Segundo, los individuos independientes tienen más probabilidades de tener nueva información en vez de los mismos datos viejos con que todos ya están familiarizados. Los grupos más inteligentes, entonces, están formados por personas con diversas perspectivas que son capaces de mantenerse independientes entre sí. (Ib., p. 41)

Al igual que en el caso de la diversidad, la independencia permite a las multitudes prevenir comportamientos desfavorables. Entre ellos está el *social proof*, que es la tendencia a asumir que si mucha gente hace algo o cree algo, entonces debe ser cierto. También está el *herding*, que ocurre por una aversión al riesgo de diferenciarse de la multitud, y lleva a sus miembros a actuar de forma de evitar el desapego del comportamiento grupal. Estos dos comportamientos son elementos característicos de los ejemplos cotidianos de que llevan a calificar a las multitudes como «estúpidas». Otro fenómeno desfavorable que señala el autor son las cascadas de información. Aquí las personas tienden a tener información diferente, imperfecta, pero igualmente valiosa. Entendiendo las decisiones de otras personas como una muestra de la información que tienen, los miembros de la multitud imitan el comportamiento de otros creyendo que aprenden algo valioso de ellos. Así, unos pocos que actúan primero

son imitados por otros, y estos a su vez, son imitados por terceros, lo que se transforma en una «secuencia de decisiones desinformadas». El problema fundamental de esto, menciona Surowiecki, «es que después de un cierto punto se vuelve racional para las personas dejar de prestar atención a su propio conocimiento —su información privada— y empezar a mirar en cambio las acciones de otros e imitarlas» (Ib., p. 54). Así, las decisiones que se toman pueden llegar a estar tan profundamente influenciadas, que la información valiosa que cada individuo aporta puede llegar a desecharse. La independencia por tanto es fundamental.

Ahora bien, a pesar de ser seres autónomos, somos también seres sociales, menciona Surowiecki. La posibilidad de generar decisiones colectivamente inteligentes se da en la base de nuestra interacción con los otros. La misma imitación que puede destruir la información privada que cada uno guarda, también puede beneficiar enormemente a la multitud. La imitación, señala el autor, permite que que no descubramos todo por nosotros mismos, y podamos beneficiarnos de los descubrimientos del resto. Es una «respuesta racional a nuestros propios límites cognitivos» (Ib., p. 58). El área de estudio es extensa, pero en lo que respecta al *wisdom of crowds*, Surowiecki señala que la imitación es una herramienta poderosa, bajo las circunstancias adecuadas.

> [El beneficio de imitar] es solo cierto en tanto las personas estén dispuestas a dejar de imitarse y aprender por sí mismas cuando los beneficios de hacerlo se vuelven lo suficientemente grandes. . . . La imitación inteligente puede ayudar al grupo —haciendo más fácil que las buenas ideas se propaguen rápidamente— pero la imitación esclavizante daña. (Ib., p. 60)

Descentralización. Esta es la tercera y última condición para que se dé el *wisdom of crowds*. La descentralización, describe Surowiecki, es lo que caracteriza a las colonias de hormigas, las colmenas de abejas, las redes sociales, las tecnologías peer-to-peer, y el Internet, entre otros. En un sistema descentralizado, señala, «el poder no reside completamente en un lugar central, y muchas de las decisiones importantes son tomadas por individuos basados en su propio conocimiento local y específico y no por un planificador omnisciente o previsor» (Ib., 70–71). El autor identifica dos elementos fundamentales en la descentralización. Primero, que fomenta, a la vez que se alimenta, de la especialización: «[La especialización] tiende a hacer a las personas más productivas y eficientes. E incrementa el alcance y la diversidad de opiniones e información en el sistema (aún cuando el interés individual de cada persona se vuelve más estrecho)» (Ib., p. 71). Segundo, que aprovecha el «conocimiento tácito», que corresponde a aquel que no puede ser transmitido fácilmente al resto pues es específico a un lugar, trabajo o experiencia particular. Esto fomenta el que las tareas sean resueltas por los más aptos. Surowiecki advierte:

> La gran fuerza de la descentralización es que alienta la independencia y la especialización por un lado al tiempo que permite a las personas coordinar sus actividades y resolver problemas difíciles por el otro. La gran debilidad de la descentralización es que no hay garantía de que la información valiosa que es descubierta en una parte del sistema hallará su camino a través del resto del sistema. (*Loc. cit.*)

Respecto al segundo punto de la cita anterior, Surowiecki indica que la descentralización sólo produce resultados inteligentes si se agrega la información de cada miembro del sistema.

Sino, el conocimiento tácito, local y privado se pierde y no puede ser aprovechado por la multitud. Para Surowiecki, el mejor ejemplo del éxito de la descentralización es el que representa Linux:

> Linux es propiedad de nadie. Cuando un problema surge con la forma en que Linux trabaja, solo llega a ser corregido si alguien, por su cuenta, ofrece una buena solución. No hay jefes ordenando por ahí a la gente, ni hay diagramas organizacionales dictando las responsabilidades de las personas. En cambio, las personas trabajan en lo que ellas están interesadas e ignoran el resto. Esto parece —y de hecho, lo es— una manera bastante desordenada de resolver las cosas. Pero hasta ahora, al menos, ha sido extraordinariamente efectiva, haciendo de Linux el retador individual más importante de Microsoft. (Ib., pp. 72–73)

[*Finaliza la revisión*]

2.3. Inteligencia colectiva

El segundo concepto fundamental dentro del contexto de máquinas sociales que estudia el potencial colectivo de grupos es el de *inteligencia colectiva*. Ese concepto, sin embargo, no es para nada nuevo, y ha sido estudiado por décadas por distintas ramas de la ciencia como son la biología, psicología y sociología, entre otras. Existen distintos enfoques al concepto; hay desde quienes la consideran como la facultad que tienen los grupos de organizarse y colaborar llegando así a resultados superiores que los individuales, hasta otros quienes la entienden como una facultad humana que permite la apertura al diálogo y comprensión del otro, a la vez que la superación del yo y la generación de nuevas formas de convivencia. Por ejemplo, en su artículo *«Colective Intelligence»*, Jan Marco Leimeister (2010) señala:

> Un enfoque ampliamente usado remonta las raíces de la inteligencia colectiva a los procesos evolutivos y se refiere a la inteligencia en grupos. En equipos deportivos y bandas musicales, e.g., cada miembro del grupo evalúa la situación general (el partido, la actuación/la música) y actúa según corresponde para alcanzar el objetivo general (ganar el partido, alcanzar una buena actuación como banda). Este comportamiento también puede ser encontrado en la fauna donde los animales se coordinan entre sí para alcanzar un objetivo común (e.g. por propósitos de caza o migración . . .). (p. 245)

En la misma línea, Jean-Francois Noubel (2008) señala:

> La inteligencia colectiva no es ni un concepto nuevo ni un descubrimiento. Es lo que da forma a las organizaciones sociales —grupos, tribus, compañías, grupos, gobiernos, naciones, sociedades, gremios, etc...— donde individuos se reúnen entre sí para compartir y colaborar, y encuentran una ventaja individual y colectiva que es más grande que si cada participante se hubiera quedado solo. La inteligencia colectiva es lo llamamos una economía de suma positiva. (p. 227)

Por otro lado tenemos enfoques mucho más sofisticados, como el de Mark Klein (2008) que sostiene que la inteligencia colectiva es «la canalización sinérgica del esfuerzo de varias mentes para identificar y llegar a un consenso sobre respuestas ante algún desafío complicado» (p. 475).

Ahora bien, el advenimiento de nuevas tecnologías de comunicación, especialmente el internet, ha permitido a un gran número de personas trabajar juntas de formas novedosas y nunca antes vistas. Esto ha abierto el campo de la inteligencia colectiva, incorporando nuevas preguntas sobre cómo las personas y los computadores pueden trabajar juntos de forma más inteligente. Una de las iniciativas más conocidas en este sentido es el Centro para la Inteligencia Colectiva del MIT[4], fundado en 2006 por Thomas Malone, actual director del centro, quien explica de la siguiente forma la orientación de las investigaciones que se realizan:

> La pregunta clave que estamos usando para organizar nuestro trabajo es: ¿cómo pueden las personas y los computadores estar conectados tal que colectivamente actúen más inteligéntemente que cualquier individuo, grupo, o computador lo haya hecho alguna vez antes? (Malone, 2008, p. 2)

La definición inicial de inteligencia colectiva con que trabajan Malone y su equipo considera que la «inteligencia colectiva son grupos de individuos haciendo cosas colectivamente que parecen inteligentes» (Íb, p. 1). Si bien esta definición es bastante amplia, en la práctica el énfasis está puesto en el nuevo tipo de inteligencia colectiva que ha aparecido en los últimos años producto de las tecnologías de la información. Entre los ejemplos que Malone destaca como nuevos tipos de inteligencia colectiva se encuentra Google, que aprovecha el conocimiento colectivo de millones de personas que hacen sitios Web para producir respuestas sorprendentemente inteligentes a las preguntas que se elaboran; Wikipedia, que usando tecnología mucho menos sofisticada, obtiene miles de personas como voluntarios en su tiempo libre para crear una increíble colección de conocimiento en línea; e InnoCentive, que permite a compañías con difíciles problemas de investigación aprovechar el conocimiento colectivo de miles de científicos partícipes de una red mundial de trabajo que ayudan a resolver esos problemas[5].

2.4. Smart mobs

El término *smart mobs*, o multitudes inteligentes, surge como un intento por describir el impacto cada vez mayor que las tecnologías de «comunicación y computación» tienen sobre la acción colectiva. El término fue acuñado por Howard Rheingold en su libro *«Smart Mobs: The Next Social Revolution»*, de 2002. Rheingold (2004) define este concepto así:

> *Smart mobs* consiste en personas que son capaces de actuar concertadamente incluso si no se conocen entre sí. Las personas que conforman *smart mobs* cooperan en formas

[4] `http://cci.mit.edu/`

[5] El lector que no conozca mayormente el caso de InnoCentive encontrará una descripción de esta iniciativa en la última sección del capítulo.

> nunca antes posible ya que portan dispositivos que poseen capacidades al mismo tiempo comunicacionales y computacionales. (p. 191)

Para Rheingold, las acciones colectivas que originan las *smart mobs* pueden consideran, por ejemplo, la interacción *peer-to-peer* como en Napster y BitTorrent, o la acción que se desarrolla en Weblogs. Incluso la Web misma, señala el autor, es un ejemplo de acción colectiva (Koman, 2003). Pero sin duda los ejemplos más emblemáticos de *smart mobs* son las manifestaciones ciudadanas auto-organizadas. A continuación las revisamos. [*Fuente de la siguiente revisión*: (Rheingold, 2004, pp. 191–193)]

Uno de los casos más emblemáticos de *smart mobs* ocurrió durante la destitución del Presidente de Filipinas, Joseph Estrada, el 20 de Enero de 2001. En medio de un escándalo político, cuenta Rheingold, Estrada pudo zafar de su juicio político gracias a sus senadores simpatizantes. Sin embargo, los residentes de Manila, la capital del país, comenzaron a manifestarse reuniéndose de a miles en la *Epifanio de los Santas Avenue* (o Edsa). Rheingold relata:

> En 75 minutos, 20.000 personas se habían reunido en Edsa, movilizados y coordinados por oleadas de mensajes de texto iniciados por líderes opositores: 'Go 2EDSA, Wear blck'. Durante cuatro días, más de un millón de personas aparecieron, mayormente vestidos de negro. Los militares quitaron apoyo al régimen; el gobierno de Estrada cayó . . . en gran parte como resultado de manifestaciones masivas no violentas. La rápida formación de la multitud anti-Estrada fue un sello de la incipiente tecnología de *smart mobs*, y los millones de mensajes de texto intercambiados por los manifestantes en 2001 fueron, según todos, una clave para el espíritu de cuerpo de la multitud. (Ib., p. 192)

A través de varios episodios similares de manifestaciones sociales, el autor evidencia el enorme potencial de acción colectiva que generan las *smart mobs*. Entre estos episodios también destaca «La batalla de Seattle» de 1999, donde se protestó en contra del encuentro anual de la World Trade Organisation (WTO). Si bien esto fue anterior al caso de Manila, aquí hubo uso de una mayor variedad de tecnologías. A través de celulares, notebooks y PDAs, los manifestantes iban actualizando constantemente páginas web que les permitían entregar y recibir reportes del estado policíaco de las calles y avenidas. Además los manifestantes se organizaban libremente según sus afinidades, yendo desde las manifestaciones no violentas hasta los intentos de arrestos masivos.

[*Finaliza la revisión*]

2.5. Crowdsourcing

El fenómeno del *crowdsourcing*[6] es por lejos uno de los más importantes que han surgido en el último tiempo, y en ello ha sido fundamental la configuración actual de la Web. También

[6]El término es una contracción de las palabras *crowd* y *outsourcing*; su traducción al español sería *externalización a la multitud*. Se preferirá el término en inglés.

es uno de los más influyentes en la noción actual de máquinas sociales y, por su propia naturaleza, comparten bastante en su significado. El término *crowdsourcing* fue acuñado por Jeff Howe en su famoso artículo *«The Rise of Crowdsourcing»* escrito para la revista Wired Magazine, y publicado en 2006. En él, Howe evidencia la aparición creciente de iniciativas en la Web que permiten a distintas clases de organizaciones realizar su trabajo aprovechando la diversa multitud conectada a través de Internet, por oposición a la pauta tradicional de contratar un grupo selecto y reducido de expertos. El autor maneja las siguientes definiciones:

> Me gusta usar dos definiciones de *crowdsourcing*:
>
> *The White Paper Version: Crowdsourcing* es el acto de tomar un trabajo tradicionalmente realizado por un agente designado (usualmente un empleado) y externalizarlo a un grupo indefinido, generalmente grande de personas en la forma de una convocatoria abierta.
>
> *The Soundbyte Version:* La aplicación de los principios Open Source a campos fuera del software. (Howe, s.f., sección Crowdsourcing: A Definition, párr. 1–3)

A continuación profundizamos esta conceptualización. Primero, revisaremos algunos casos de *crowdsourcing*. Esos casos corresponden a los principales ejemplos que menciona Howe en su artículo inaugural, y que le llevaron a la introducción del concepto. Luego estudiaremos las razones que Howe identifica como fundamentales para la emergencia de este fenómeno. Posteriormente, dirigiremos la atención a comprender la diversidad de versiones que ha recibido el concepto.

Partamos por iStockphoto[7]. Esta iniciativa se enmarca en el contexto del mercado de fotografías profesionales. En general, cuando una persona o institución —e.g. diseñador, director artístico, museo— necesita conseguir fotografías de alta calidad, se ve enfrentado a los problemas del *copyright* y las elevadas sumas de dinero que hay que desembolsar. Las opciones generalmente son contratar un fotógrafo profesional, o bien utilizar imágenes preexistentes, lo que se conoce como fotografía de stock. Aquí los fotógrafos profesionales reúnen colecciones de imágenes, y las venden luego de acuerdo a las necesidades del cliente. El problema son los altos costos de estas fotografías. iStockphoto nació por iniciativa de un grupo de diseñadores Web que en 2000 creó un sitio para poder usar y compartir libremente sus propias fotografías, con el fin de evitar el costo de las fotografías de stock. Cada vez más personas se integraron al sitio, y así se empezó a crear una comunidad de fotógrafos amateurs en torno a iStockphoto. Los dueños decidieron cobrar una pequeña cuota por el uso de las imágenes, que se reparte entre ellos y el fotógrafo correspondiente. Así se originó iStockphoto, organización que hoy día representa una fuerte competencia al mercado de fotografías de stock. Trabajan con una multitud diversa de fotógrafos amateurs, cuya motivación principal es el gusto por la fotografía, y que aún así satisfacen las necesidades de calidad de los clientes. La idea es que esos clientes se ven enfrentados a un material diverso, original y a un bajo costo. En su artículo de 2006, Howe subrayaba el rápido crecimiento que iStockphoto había alcanzado. A solo seis años de iniciar un sitio para compartir material entre colegas, tenían más de 22.000 contribuyentes, la cuota por una fotografía básica era de $1 a $5 dólares —versus los $200 a $300 de la fotografía de stock—, y habían vendido más de 10 millones de imágenes, varias veces más que lo que la principal empresa de fotografías de stock había vendido (Howe, 2006).

[7] (`http://www.istockphoto.com`)

Otro caso que se menciona es el de InnoCentive[8]. Esta compañía utiliza el *crowdsourcing* para reunir una extensa red de científicos en busca de problemas que resolver. Los clientes de InnoCentive son empresas de distintos rubros, que se ven enfrentadas a resolver problemas de carácter científico como parte del área de investigación y desarrollo (R&D). InnoCentive toma el problema de cada cliente, y lo pone a disposición de la red de científicos. Una vez que se encuentra una solución a un problema, la persona que lo resolvió recibe una recompensa monetaria que depende de cada problema.

> Los clientes de InnoCentive incluyen firmas de [la lista] Fortune 500 como Procter & Gamble (P&G), DuPont, y BASF. Cuando los equipos internos de R&D de las compañías no pueden resolver un problema, ellos lo entregan a los 140.000 científicos de más de 170 países que exploran regularmente el sitio Web de InnoCentive en busca de trabajo. La mayoría de las recompensas por soluciones exitosas pagan entre $10000 y $100000. (Howe, 2008, p. 42)

Los problemas que los clientes proponen a InnoCentive son bien específicos, como señala Howe. Crear colorantes para indicar la concentración correcta de detergente, encontrar un biomarcador para tratar la esclerosis ALS, e inyectar fluoruro en un tubo de pasta de dientes sin dispersarlo, son algunos de ellos (Howe, 2006, 2008). InnoCentive logra reunir a muchas personas con formación científica que no practican ciencia y que en sus tiempos libres pueden encontrar una forma de hacer lo que les gusta. Las personas en la red trabajan desde sus hogares utilizando «laboratorios caseros». Hay desde amateurs hasta Ph.D., y pueden trabajar no sólo en problemas relacionados en su área, sino en cualquiera de los desafíos que los clientes de InnoCentive propongan.

El último caso que mencionaremos es el de Mechanical Turk[9] de Amazon. Esta es una plataforma Web similar a InnoCentive, en el sentido que conecta trabajadores con solicitantes para resolver problemas que requieren de la inteligencia humana. Pero se diferencia en que el trabajo se presenta en la forma de pequeñas tareas, realizables en poco tiempo, y para las que no se necesita una formación específica. Este tipo de servicio, que se conoce con el nombre de *microtasking*, implica poder dividir un trabajo en pequeñas tareas de modo que al agregar sus soluciones se resuelve el problema original.

> Mechanical Turk de Amazon es un mercado basado en la Web que ayuda a compañías a encontrar personas para realizar tareas en las que los computadores son generalmente pésimos —identificar ítemes en una fotografía, revisar documentos de bienes raíces para encontrar información de identificación, escribir pequeñas descripciones de productos, transcribir podcasts—. Amazon llama a las tareas HITs (tareas de inteligencia humana); están diseñadas para requerir muy poco tiempo, y consecuentemente ofrecen muy poca compensación —la mayoría de unos pocos centavos a unos pocos dólares—. (Howe, 2006, p. 4).

Mechanical Turk se presenta como una plataforma de *crowdsourcing* completamente genérica. Junto a los ejemplos ya citados, otros casos de HITs son el seleccionar la ortografía

[8] `http://www.innocentive.com`
[9] `https://www.mturk.com`

correcta para términos de búsqueda, calificar los resultados de una búsqueda, calificar si un sitio Web es apropiado para la audiencia general, verificar si dos productos son el mismo, traducir un párrafo del inglés al francés, y categorizar productos, entre otros (Amazon Mechanical Turk, s.f., sección Examples, párr. 2). El acto de resolver este tipo de tareas se conoce con el nombre de *computación humana*. Este concepto es muy importante dentro del área de *crowdsourcing*, y por lo tanto también en el de máquinas sociales. En general, la computación humana se entiende como el uso de humanos para solucionar problemas difíciles o imposibles de resolver para los computadores, pero fáciles para las personas. Volveremos a ello más adelante.

En su libro *«Crowdsourcing»* de 2008, Howe identifica y analiza cuatro desarrollos fundamentales que hicieron al *crowdsourcing* «no solo posible, sino que inevitable». Estos pilares que dieron vida al fenómeno que atendemos son: el «renacer» del amateurismo, la emergencia del movimiento open source, la disponibilidad creciente de herramientas de producción, y el nacimiento de comunidades en línea organizadas en torno al interés de las personas. A continuación revisamos brevemente estas ideas. [*Fuente de la siguiente revisión*: (Howe, 2008, pp. 23–127)]

Amateurismo. La pasión amateur representa para Howe «el combustible del motor del *crowdsourcing*». En general, plantea, las personas que participan en proyectos de *crowdsourcing* no tienen como motivación principal el dinero, y el trabajo lo hacen en sus horas de ocio, entregándose a algo que aman hacer. Desde su punto de vista los amateurs son la fuerza de trabajo genuina del *crowdsourcing*. Howe analiza cómo el amateurismo ha visto un verdadero «renacer» en el último tiempo. Entre las principales razones que nombra está el aumento exponencial de la educación y la accesibilidad al conocimiento que trajo el Internet. Así se forma una fuerza de trabajo talentosa y altamente calificada, pero que lamentablemente se enfrenta a un mercado que tiene necesidades distintas, o requiere incluso más grados de especialización. Las personas por lo tanto tienen una educación más allá de la que necesitan para su trabajo, y se sienten poco realizadas. A lo anterior, por supuesto, hay que agregar los anhelos e intereses más profundos que guardan las personas, así como los hobbys y gustos personales, y aquellos que aún les queda por descubrir. Eso ha producido un creciente aumento de la actividad amateur —que muchas veces tiene estándares profesionales, explica el autor—, y es ese el factor que aprovecha el *crowdsourcing*.

Howe hace también una reflexión sobre este concepto de amateurismo. El término proviene del latín *amare*, que significa amar. Ser amateur ha llegado a tener la connotación hoy en día de ser principiante, primerizo o inexperto. Sin embargo, explica el autor, el amateurismo es lo que impulsó la propia ciencia en la Inglaterra del siglo XVII, o sea, los tiempos de la Royal Society. La aristocracia aborrecía la profesión, por ver el ganar dinero a través del trabajo como algo inferior, y creían en la búsqueda del conocimiento por sí mismo. La ciencia se desarrolló por mucho tiempo en la forma del caballero erudito amante de la ciencia, el ideal amateur. Todos los avances fundamentales que se dieron en la ciencia en ese período, se dieron precisamente en la forma del amateurismo. La historia es por supuesto extensa, y continúa hasta llegar a la actual profesionalización de la ciencia. El sentido de la reflexión de Howe es bastante importante, porque el amateurismo que alimenta al *crowdsourcing* no es una calificación de profesionalismo o no; ambos tipos de *credenciales* existen en el *crowdsourcing*. Lo que sí distingue a quienes participan de éste, es el amor a lo

que hacen. En este mismo sentido, un claro ejemplo de la capitalización que el *crowdsourcing* hace del amateurismo es eBird[10]. Esta plataforma, creada en 2002, permite a los amantes de las aves vivir la experiencia de realizar observaciones de campo cuyos resultados aportan significativamente a la ornitología. Reúne a personas con pasión de avistar aves, identificar cantos y distinguir especies. Las personas suben listas de avistamiento de aves que encontraron ya sea en una excursión preparada, al pasar por una plaza, o al ir de regreso a casa. Luego esos datos van a la Avian Knowledge Network, una base de datos con la que pueden trabajar los ornitólogos profesionales. En 2003 se recolectaron 100.000 avistamientos, y para el 2008 la cifra ascendió a 1,15 millones. Howe subraya el impacto amateur en esta disciplina: «Las enormes cantidades de datos siendo generados por ornitólogos amateurs están proveyendo una mirada sin precedentes a los patrones migratorios y de distribución de una multitud de especies» (Ib., p. 31).

Movimiento open source. Desde el punto de vista del autor, si el amateurismo representa el combustible del *crowdsourcing*, entonces el movimiento de software open source es el molde. Howe señala:

> A principios de los 1990s, la multitud había producido su primer trabajo sustancial: Linux, un sistema operativo superior, en muchos aspectos, a los mejores productos de cualquier corporación. El open source proporcionó un precedente —una prueba de concepto—. Si las personas trabajando en su tiempo libre —químicos de cocina, músicos de sótano, fotógrafos de Domingo— proveen al motor del *crowdsourcing* de combustible, es el movimiento de software open source el que lo dotó de un molde. (Ib., p. 48)

En un comienzo, relata el autor, todo el código que se escribía era código abierto. Este se desarrollaba principalmente en la academia, y compartir el código era tanto una muestra del tradicional intercambio de conocimiento, como la única forma de afrontar los costos y dificultades de programar en los primeros computadores. Así nació una cultura de programadores colaborativa, creativa y abierta a compartir su trabajo. No era fácil separarlos de los usuarios, porque ellos eran los principales usuarios, señala Howe. Eran los *hackers* originales. Fue con la introducción del computador personal que el software propietario entró en juego, cuando Bill Gates y Paul Allen comenzaron a licenciar y privatizar el código, relata el autor. Pero en 1983 un joven científico decide hacer la guerra al software propietario. Era Richard Stallman, quien fundó el proyecto GNU como un esfuerzo por crear un sistema operativo basado en código abierto y libremente distribuido —en ese momento hasta Unix se había vuelto propietario, por eso el acrónimo recursivo de GNU (GNU is Not Unix)—. Lo que Stallman hizo fue también un molde para los sistemas de *crowdsourcing*. Al basarse en Unix, el sistema trabajaba con miles de pequeños archivos, pequeñas tareas que otros podían elegir para desarrollar según su disponibilidad. Esta subdivisión del trabajo estuvo acompañada de la libertad para copiar, editar y lo más importante, añadir las contribuciones que cualquiera quisiera hacer. En 1985, Stallman creó la Free Software Foundation (FSF) para promover el software libre. En 1986 creó un compilador para C basado únicamente en código libre, y para que nadie lo mal utilizara, creó la GNU General Public License, lo que según Howe es una de las más grandes contribuciones al movimiento open source. Básicamente si algo es liberado bajo esta licencia entonces puede ser usado libremente, y si se quiere utilizar como parte

[10] `http://ebird.org/`

de otro producto, este debe usar la misma licencia —concepto conocido como *copyleft*—. El problema de Stallman era que aún no podía escribir el kernel del sistema operativo. Hasta que en 1991 aparece Linus Torvalds, y hace una convocatoria abierta para ayudar en el desarrollo del kernel en que estaba trabajando. En dos años, miles de programadores ayudaban a mejorar Linux, señala Howe. El surgir de este exitoso movimiento open source dibujó también los planos del *crowdsourcing*.

> ¿Qué hace al open source tan eficiente? En el más amplio de los sentidos, es la capacidad para que un gran número de personas contribuyan. El evangelista del open source Eric S. Raymond resumió con fama esta verdad fundamental cuando escribió, que, "Dado suficientes ojos, todos los errores son superficiales" —lo que es decir que ningún problema es muy difícil si suficientes personas intentan atacarlo—. Poniéndolo de otra forma, una fuente laboral grande y diversa consistentemente llegará a mejores soluciones que la fuerza de trabajo más talentosa y especializada. Esto es cierto en campos como ciencia corporativa, diseño de productos, y creación de contenido como lo es en el software, y es uno de los principios centrales del *crowdsourcing*. (Ib., p. 54)

Howe destaca además tres factores fundamentales del movimiento de software open source que se extrapolaron a otras áreas fuera de la computación: «Primero, una tarea enorme distribuida a través de una red masiva. Segundo, no hay límite en el número de contribuyentes potenciales. Finalmente, el trabajo mismo es fracturado en tareas pequeñas y discretas» (Ib., p. 62).

Herramientas de producción. La creciente disponibilidad de los medios de producción es lo que ha permitido a la multitud empoderarse para ejercer su amateurismo y competir en mayor igualdad de condiciones en dominios que antes pertenecían a unos pocos. Aunque haya un gran amor por la fotografía, y aunque exista una gran red de fotógrafos amateurs, proyectos de *crowdsourcing* como iStockphoto no tienen sentido si los dispositivos son inaccesibles económicamente para las personas, y si la información sobre cómo realizar capturas, editar y publicar fotos fuera inexistente. La creciente superación de ese tipo de problemas es lo que permite a las personas desarrollar las actividades que quieren con un nivel cada vez mayor de profesionalismo. Esto es lo que permite a la multitud participar de muchas iniciativas de *crowdsourcing* que no podrían hacerse si no se tuvieran los medios ni la instrucción adecuada.

Howe realiza un análisis de por qué los medios de producción se han vuelto cada vez más accesibles a las personas. El primer factor que reconoce es la caída del precio de las herramientas mismas de producción. También estas herramientas se vuelven cada vez más fáciles de usar. Junto a la caída en los precios de las cámaras digitales profesionales de video, también nace nuevo software que permite la edición y producción del material, y que facilita el trabajo enormemente. Como menciona Howe, lo que antiguamente se hacía cortando y juntando manualmente tiras de celulosa, hoy se hace a través de software amigable, rápido de aprender y sencillo de usar —y en el que equivocarse es una acción fácilmente reversible—. Otro factor que hay que mencionar es el creciente acceso a la información sobre cómo usar las herramientas de producción. La Web está llena de tutoriales para aprender a usar todo tipo de productos, y además es posible educarse en distintas áreas a través de sitios aficionados y donde incluso llegan a participar profesionales. Más aún, existe una serie de nuevas plataformas como Coursera y iTunes U que ofrecen por la Web cursos universitarios

abiertos a todo público. Finalmente, gracias al Internet, el costo de distribuir el trabajo es prácticamente nulo. «La tecnología se mueve», señala Howe, «en direcciones sencillas: más barato, más rápido, más pequeño, y más fácil de usar» (Ib., p. 78).

Comunidades en línea. El último factor que Howe identifica como fundamental para el nacimiento del *crowdsourcing* es la emergencia de comunidades en línea en torno a los propios intereses personales. Para el autor, la comunidad es la «fuerza organizadora básica detrás del *crowdsourcing*» (Ib., p. 100). En el pasado, explica Howe, las comunidades se formaban de acuerdo a las posibilidades geográficas. La creación del Internet y el nacimiento de la Web permitieron que las personas ahora puedan organizarse de acuerdo a líneas de afinidad. Otras consideraciones, señala Howe, como la geografía, las clases sociales o el nivel educacional, ya no tienen importancia. Por su propia naturaleza, los grupos formados entorno a líneas de interés están «adaptados de forma única a la tarea de producción de información» (Ib., p. 120). Además, las comunidades juegan el rol de generar incentivos para que sus miembros se sientan más motivados, acogidos y encuentren un lugar donde perfeccionarse. Las motivaciones monetarias, señala Howe, están muy por debajo de las motivaciones intrínsecas como la realización personal, el sentimiento de equipo, el desarrollo de un proyecto en común, la construcción de una reputación y el reconocimiento social. La comunidad provee un contexto para que estos incentivos existan. Howe concluye:

> Cuatro desarrollos crearon el terreno fértil en el que el *crowdsourcing* podría emerger. El surgir de una clase amateur estuvo acompañado por la emergencia de un modo de producción —software open source— que proporcionó inspiración y dirección práctica. La proliferación del Internet y herramientas baratas dieron a los consumidores el poder antes restringido a las compañías dotadas con vastos recursos de capital. Pero fue la evolución de las comunidades en línea —con su capacidad de organizar eficientemente a las personas en unidades económicamente productivas— lo que transformó los primeros tres fenómenos en una fuerza irrevocable. (Ib., p. 99)

[*Finaliza la revisión*]

Con el paso del tiempo, el *crowdsourcing* ha recibido una intensa atención por parte de la academia, y se han desarrollado distintas versiones del significado original. Mientras algunas aproximaciones buscan reducir el sentido del concepto, otras han hecho lo contrario, generalizándolo a la vez que ensanchando fuertemente el repertorio de casos de *crowdsourcing*. Entre este tipo de enfoques más genérico destaca el trabajo de Doan, Ramakrishnan y Halevy (2011), quienes consideran como *crowdsourcing* a una amplia gama de casos que incluyen desde los ejemplos más tradicionales hasta algunos sumamente novedosos. Encontramos a Linux, Wikipedia, Youtube, Flickr, Yahoo! Answers, eBay, World of Warcraft, el buscador de Google, el corrector ortográfico de Google, el recomendador de Amazon y los Iowa Electronic Markets, entre otros. El trabajo de los autores es característico por dos razones: la primera, por proponer un enfoque genérico desde el que observar distintos sistemas Web; la segunda, porque los autores elaboran una detallada clasificación de las aplicaciones de *crowdsourcing*, lo que ayuda a reconocer e identificar con mayor claridad las similitudes y diferencias entre los distintos casos que hoy existen en la Web. Los autores trabajan uniformemente con el término «sistemas de *crowdsourcing*» para referirse a lo que nosotros hemos estado llamando iniciativas, aplicaciones o proyectos de *crowdsourcing*. La definición propuesta por Doan et

al. (2011) es la siguiente:

> Decimos que un sistema es un sistema de CS [crowdsourcing] si enlista una multitud de humanos para ayudar a resolver un problema definido por los dueños del sistema, y si haciéndolo, enfrenta los siguientes cuatro desafíos fundamentales: ¿Cómo reclutar y retener usuarios? ¿Qué contribuciones pueden hacer los usuarios? ¿Cómo combinar las contribuciones de los usuarios para resolver el problema objetivo? ¿Cómo evaluar a los usuarios y sus contribuciones? (pp. 87-88)

Los autores destacan que estos desafíos no son característicos de todo sistema centrado en humanos. Es el caso de lo que ellos denominan *crowd managment systems* y que ejemplifican mediante los sistemas de control de tráfico vehicular. Allí la reclusión de más usuarios es un hecho que se quiere evitar, en oposición a algo que se quisiera fomentar. Los autores realizan una detallada categorización de los principales elementos que intervienen a la hora de caracterizar los sistemas de *crowdsourcing*. El resultado considera nueve «dimensiones» transversales al *crowdsourcing*, y la correspondiente identificación de las propiedades fundamentales de estos sistemas desde la perspectiva de cada dimensión. En algunas dimensiones estas propiedades se presentan como categorías que permiten clasificar todo el espectro de sistemas, mientras que en otras se presentan como una formulación de los elementos más importantes a considerar y tener en cuenta. Esta categorización es importante no sólo porque permite reconocer elementos distintivos de cada clase de sistema de *crowdsourcing*, sino que también por reflejar la dificultad de encontrar y formular categorías perfectamente ajustadas y evidentes. Las revisamos a continuación. [*Fuente de la siguiente revisión*: (Doan et al., 2011, pp. 88–96)]

Naturaleza de la colaboración. Ésta puede ser *implícita* o *explícita*. En una colaboración implícita, los usuarios no contribuyen intencionadamente para resolver el problema, sino que lo hacen como efecto secundario de otra acción distinta. Ejemplo de ello son los GWAPs, donde los usuarios buscan primeramente entretenerse[11]. En oposición, una colaboración explícita corresponde a una acción cuyo propósito principal es contribuir. Un ejemplo tácito de esto es la edición de una entrada en Wikipedia.

Tipo de problema objetivo. Éste puede ser «cualquier problema definido por los dueños del sistema (por ejemplo, construir artefactos temporales o permanentes, ejecutar tareas)» (Ib., p. 88). En el primer caso está Linux; en el segundo, Mechanical Turk de Amazon. En general la clasificación de esta dimensión no es exhaustiva ni intenta cubrir todo el espectro posible; en cambio, los autores dan ejemplos bastante concretos de clases de problemas, como evaluar una colección de ítemes, etiquetar imágenes, predecir eventos, digitalizar texto escrito, y recomendar productos, entre otros.

Grado de esfuerzo manual. Corresponde al esfuerzo manual ejercido al trabajar en los cuatro desafíos fundamentales. Puede ser muy simple, como valorar un libro en Amazon, hasta muy complejo, como escribir una contribución para el servidor Apache. Los autores también señalan que se debe tener en cuenta el cómo dividir el esfuerzo manual entre

[11] *Games With A Purpose*, o juegos con propósito, son juegos de entretención en línea donde la gente resuelve tareas difíciles de realizar para computadores pero fáciles de realizar para humanos, como efecto colateral de entretenerse jugando.

usuarios y los dueños del sistema. Un caso de esfuerzo cargado hacia los dueños es la detección de usuarios maliciosos, explican los autores. Aquí los usuarios clickean un perfil o entrada como sospechosa, pero la revisión y evaluación de ello recae en los dueños. En el caso contrario de esfuerzo cargado hacia los usuarios se encuentra, nuevamente, la edición en Wikipedia.

Rol de usuarios humanos. Se consideran cuatro roles no exclusivos: el rol de *esclavo* se desempeña al resolver problemas mediante la lógica «dividir para reinar»; el *proveedor de perspectiva* contribuye con su propia perspectiva y que al ser combinada con la de otros producen una solución mejor que la mera suma de las partes; el *proveedor de contenido* contribuye con contenido autogenerado;y el *proveedor de componente* funciona como componente «en un artefacto objetivo, como una red social o simplemente una comunidad de usuarios (de modo que el dueño pueda, por ejemplo, vender anuncios)» (Ib., p. 89).

Arquitectura. Esta puede ser *autónoma* o *dependiente*[12]. El criterio que se usa para diferenciar el tipo de arquitectura es si el sistema de *crowdsourcing* enfrenta todos los desafíos fundamentales o no. Si lo hace, es autónomo, sino, dependiente. En el primer caso, tenemos a Facebook, en el segundo tenemos al corrector ortográfico de Google, al recomendador de libros de Amazon y al Yahoo's Search Assist. La razón de considerar sistemas de *crowdsourcing* que no resuelven todos los desafíos es porque pueden depender de aplicaciones más grandes que les solucionan algunos de ellos —típicamente la dependencia se da en la forma de servicios embebidos o subsitemas—. El caso emblemático son los sistemas que explotan la traza que los usuarios dejan en sistemas asociados, como los ya mencionados. En ellos, el sistema dependiente no se ve enfrentando a reclutar usuarios, ni decidir los tipos de contribución. Aún así, sí se ve enfrentado a combinar y evaluar las contribuciones.

Reclutamiento y retención de usuarios. Para la reclusión, los autores proponen cinco soluciones principales: se les puede exigir contribuir si se tiene la autoridad, como en el caso de algún sistema interno de una compañía; se les puede pagar como en Mechanical Turk; se puede convocar voluntarios como en Wikipedia y YouTube; se puede hacer a los usuarios pagar por un servicio, si para permitirles acceder a un cierto servicio se les exige contribuir en otro. Un excelente ejemplo de esto lo constituye reCAPTCHA, sistema al que muchas personas contribuyen como medio para acceder a un servicio completamente distinto. Finalmente, se puede aprovechar la traza de usuarios, como vimos en el punto anterior. Para la retención de usuarios, los autores señalan como las estrategias más populares la gratificación instantánea; entregar una experiencia agradable o servicio necesario; jugar con la fama, confianza o reputación; trabajar con los conceptos de usuarios mejor evaluados; y finalmente, generar sentimiento de propiedad sobre elementos del sistema.

Contribución de usuarios. En general los usuarios pueden *evaluar* —libros, películas, entre otros— con comentarios, puntajes o tags; *compartir* ítemes como productos, servicios, o conocimiento textual —Youtube, CPAN, Twitter—; *integrarse a redes sociales* contribuyendo al crecimiento colaborativo de grafos sociales en términos de nodos y arcos —Facebook—; *construir artefactos* —Apache, Linux, Wikipedia—; *ejecutar tareas* —GalaxyZoo, Mechanical Turk—; y *contribuir implícitamente* como en GWAPs o en jue-

[12]Los términos originales del inglés son *standalone* y *piggyback*.

gos multiusuarios masivos como World of Warcraft. Además los autores identifican cuatro elementos a considerar a la hora de pensar en las posibles contribuciones de los usuarios: ¿Qué tan demandantes cognitivamente son las contribuciones? ¿Cuál debería ser el impacto de una contribución? ¿Cuál es la contribución de la máquina? Y ¿qué tan fácil para los usuarios es contribuir a través de la interfaz de usuario?

Combinación de contribuciones. Muchos sistemas no combinan las contribuciones, o lo hacen sin mayor elaboración, señalan los autores. Por ejemplo, los sistemas de evaluación de libros no combinan los comentarios o revisiones, y los ratings son calculados con fórmulas simples. Del mismo modo, las redes sociales simplemente enlazan las contribuciones para formar el grafo de red social. Sin embargo hay sistemas más complejos donde las contribuciones están más acopladas como los juegos y los sistemas que construyen software, entre otros. Independientemente de lo anterior, un problema central que destacan los autores es saber qué acción tomar cuando algunos usuarios dicen una cosa y otros la niegan. Algunas soluciones automáticas, por ejemplo, combinan los resultados ponderándolos con la reputación de los usuarios. En las soluciones manuales, en cambio, son los usuarios los que deben resolver. Aquí muchas soluciones usan las estructuras jerárquicas que hay entre los miembros para zanjar las discusiones.

Evaluar usuarios y contribuciones. En este punto los autores centran la atención principalmente en cómo detectar, disuadir y bloquear usuarios maliciosos. Primero, para evitar sus contribuciones, se puede limitar el tipo de la contribución según el tipo de usuario. Segundo, para detectar tanto usuarios maliciosos como sus contribuciones, se puede monitorear el sistema por parte de los dueños, distribuir el monitoreo a una parte de usuarios confiables, permitir a los usuarios ordinarios dar aviso al sistema, o también calcular la confiabilidad de usuarios en base a preguntas con respuestas conocidas, entre otros. Finalmente, se puede disuadir a usuarios maliciosos con distintos castigos desde el baneo hasta la vergüenza pública. También se debe considerar la acción opuesta, es decir identificar a los usuarios productivos e influyentes para promoverlos y asignarles mayor responsabilidad.

[*Finaliza la revisión*]

Como se mencionó anteriormente, el trabajo de Doan et al. se enmarca en un amplio conjunto de enfoques que la academia ha generado respecto del concepto de *crowdsourcing*. En particular, debido a las distintas aproximaciones que los investigadores han desarrollado se ha acumulado un considerable repertorio de definiciones. En 2013, Hetmank condujo una revisión sistemática de la literatura sobre el tópico de *crowdsourcing* con la finalidad de «obtener una mejor comprensión de qué son los sistemas de *crowdsourcing* y qué aspectos típicos de diseño son considerados en el desarrollo de tales sistemas» (p. 55). El trabajo realizado por el autor es bastante esclarecedor sobre la realidad del concepto de *crowdsourcing*. Lo revisamos brevemente a continuación. [*Fuente de la siguiente revisión*: (Hetmank, 2013, pp. 57–65)]

Metodológicamente el estudio considera una pre-selección de 1699 artículos científicos publicados en *journals* y *conference proceedings*, como resultado de aplicar el termino búsqueda «*crowdsourcing*» en las principales bases de datos de papers académicos relevantes al tema[13].

[13] A saber, ACM Digital Library, Ebscohost, Emerald, IEEE Xplore Digital Library, SAGE Journals, Scien-

De la pre-selección se filtraron solo las entradas en que *crowdsourcing* aparecía como *keyword*, generando un total de 337. Luego, de estas últimas entradas se escogió sólo las que respondieron a los términos de búsqueda «*crowdsourcing systems*», «*crowdsourcing application*», o «*crowdsourcing platform*», llegando a un total de 220 artículos que conformaron el universo a revisar. Como resultado de la investigación, Hetmank identificó 17 tipos distintos de definiciones relacionadas a los tres términos mencionados, los cuales agrupó en cuatro clases o «perspectivas» distintas. Estas son:

Perspectiva organizacional. Resalta el papel de los sistemas de *crowdsourcing* como distribuidores de «tareas que son emitidas por un solicitante (dueño del sistema, empleador) a los potenciales destinatarios (multitud, trabajadores humanos)» (Ib., p. 61).

Perspectiva técnica. Se centra en los aspectos técnicos de los sistemas de *crowdsourcing* —API, interfaces de usuario, autentificación de usuarios, seguimiento del historial, mecanismos de pago, entre otros—.

Perspectiva de proceso. «Detalla acciones que usualmente se realizan sobre objetos de datos y usuarios» (*Loc. cit.*). Es la visión del sistema como «caja blanca». Ejemplo de estas acciones son la división y asignación de tareas, la combinación de contribuciones, entre otros.

Perspectiva centrada en el humano. Se centra en los conceptos de inteligencia colectiva e interacción humana en tanto conductores principales del *crowdsourcing*.

Hetmank deriva además resultados respecto a los principales elementos de diseño considerados al desarrollar sistemas de *crowdsourcing*. Identifica cuatro. Primero, un componente para la *administración de usuarios*, que debe considerar funciones para el registro, evaluación, agrupación y coordinación de usuarios. Segundo uno para la *administración de tareas*, que debe proveer funciones para el diseño y asignación de tareas. Tercero, un componente para la *administración de contribuciones*, que debe considerar funciones para la evaluación, preproceso y combinación de contribuciones, así como para la selección de la solución. Y por último, un componente para la *administración del workflow*, que debe permitir definir y administrar el *workflow*, entendiendo que se requieren generalmente varias iteraciones para encontrar un *workflow* de *crowdsourcing* eficiente.

[*Finaliza la revisión*]

ceDirect, Springerlink, y Wiley.

Capítulo 3

Principales enfoques en el estudio de máquinas sociales

En este capítulo terminaremos de explorar la noción de máquinas sociales revisando los principales enfoque académicos con que se ha estudiado hasta la fecha este concepto. Comenzamos revisando el enfoque que da la ciencia Web, que posiciona la noción de máquinas sociales como pieza clave para el entendimiento de la Web misma. Luego examinamos una aproximación importante en la búsqueda de los elementos constitutivos de las máquinas sociales. Después veremos una aproximación desde el área de Adquisición de Conocimiento a la noción de máquinas sociales, y posteriormente estudiaremos un framework para clasificar máquinas sociales que es el resultado de uno de los trabajos de investigación más relevantes en el área de los últimos años. Para finalizar presentaremos una reciente investigación que elabora una visión más general de máquinas sociales, englobando los distintos enfoques con que hasta hoy se ha estudiado esta noción.

3.1. Las máquinas sociales y la ciencia Web

Uno de los principales enfoques que ha recibido la noción de máquina social es el que entregaron Hendler, Shadbolt, Berners-Lee y Weitzner en su artículo *«Web Science: An Interdisciplinary Approach to Understanding the Web»*, de 2008. Allí, Hendler et al. incluyen el tópico de máquinas sociales dentro de los principales desafíos en investigación de la emergente área llamada *Web science*, o ciencia Web. Ésta busca integrar el estudio de la Web bajo un único tópico interdisciplinario que reúna las investigaciones que tradicionalmente se han estado desarrollando bajo distintos formatos —social, comunicacional, algorítmico, entre otros—. A continuación revisamos el enfoque que estos autores ofrecieron sobre el concepto de máquinas sociales. [*Fuente de la siguiente revisión*: (Hendler et al., 2008, pp. 63–67)]

Hendler et al. analizan la Web desde la perspectiva de una microescala y una macroescala. En la microescala, la Web es un conjunto de protocolos y lenguajes desde los cuales se construyen las aplicaciones. El uso de la microescala por un número cada vez mayor de usuarios

genera fenómenos a macroescala que caracterizan el tipo de comportamiento e interacción social de las aplicaciones. Bajo este contexto, Hendler et al. destacan que para entender la Web es esencial llegar a comprender cómo predecir los efectos a macroescala a partir del diseño a microescala, y cómo poder construir exitosamente desde esa microescala sistemas que en la macroescala se comporten como se quiere. Para los autores el «modelo social» que se produce por las interacciones sociales que permite una tecnología determinada, no se puede explicar solo en términos de esa tecnología. Entonces Hendler et al. hacen uso de la perspectiva de máquinas sociales para así observar los sistemas Web no sólo como tecnologías, sino también como sistemas sociales, permitiendo comprender de mejor forma la relación entre las características tecnológica y las consecuencias sociales.

> La idea de una máquina social fue introducida en *Weaving the Web*, la que dio como hipótesis que el diseño arquitectónico de la Web permitiera a desarrolladores, y por tanto a usuarios finales, usar la tecnología computacional para ayudar a proveer la función administrativa de sistemas sociales mientras fueran realizados en línea. La máquina social incluye la tecnología subyacente (mediaWiki en el caso de Wikipedia) pero también las reglas, políticas, y estructuras organizacionales usadas para administrar la tecnología. Ejemplos abundan en la Web hoy. Considérese el acoplamiento del diseño de aplicaciones de sistemas de soporte de blogs (como LiveJournal y WordPress) con los mecanismos sociales proporcionados por *blogrolls*, *permalinks* y *trackbacks* que han dado lugar a la llamada blogósfera. Similarmente, los protocolos usados por sitios de redes sociales como MySpace y Facebook tiene mucho en común, pero el éxito o fracaso de los sitios depende de las reglas, políticas, y de las comunidades de usuarios que soportan. Dado que el éxito o fracaso de las tecnologías Web generlamente descansa en estas características sociales, la capacidad de diseñar aplicaciones exitosas requiere un mejor entendimiento de las características y funciones de los aspectos sociales del sistema. (Ib., p. 66)

Desde este punto de vista las máquinas sociales son una noción fundamental en el estudio de la Web. Hendler et al. reconocen también que hoy en día los sistemas se comportan como «máquinas sociales muy tempranas», debido al aislamiento y la falta de interacción entre ellas. En ese sentido los autores manejan la hipótesis de que en el futuro las máquinas sociales serán más efectivas, estarán enlazadas, al igual que los procesos sociales, y deberán sustentarse en tecnologías que permitan a cada comunidad de usuarios crear y usar sus propias máquinas. Hendler et al. señalan, no obstante, que antes que una nueva generación de máquinas sociales pueda emerger, deben solucionarse una serie de desafíos de investigación en máquinas sociales. Estos son determinar los fundamentos teóricos de máquinas sociales, reconocer los principios de arquitectura necesarios, explicitar las políticas sobre compartir información garantizando su uso acorde a las expectativas sociales, y entender el rol que las diferencias culturales juegan en la Web.

[*Finaliza la revisión*]

3.2. Las máquinas sociales y sus elementos constitutivos

Meira et al. (2010) realizaron un gran avance en cuanto al estudio de máquinas sociales como unidades autocontenidas e interconectables. Los autores enfocan el estudio de máquinas sociales desde la perspectiva de identificar los elementos básicos que permiten describir tanto las funcionalidades de como las interacciones entre máquinas sociales. En esa dirección postulan una noción algebraica de máquinas sociales, donde la definen como una tupla de componentes. Meira et al. (2010) señalan:

> En general una Máquina Social (SM) representa una entidad conectable conteniendo una *unidad de procesamiento* interna y una interfaz cobertora que espera por *requerimientos* desde y replica (con *respuestas*) hacia otras máquinas sociales. Su unidad de procesamiento recibe *inputs*, produce *outputs* y tiene *estados*; y sus *conexiones* definen relaciones intermitentes o permanentes con otras SMs, conexiones que están establecidas bajo conjuntos específicos de *restricciones*. . . . Nosotros definimos una SM como una tupla de *Relaciones, Interfaz Covertora, Requerimiento, Respuesta, Estado, Restricciones, Input, Unidad de Procesamiento, y Output*, como sigue: $SM =< Rel, WI, Req, Resp, S, Const, I, P, O >$. (p. 4)

En la Figura 3.1 se reproduce el diagrama que Meira et al. utilizaron para explicar su noción de máquinas sociales. A continuación revisamos los elementos propuestos en esta definición. [*Fuente de la siguiente revisión*: (Meira et al., 2010, pp. 4–6)]

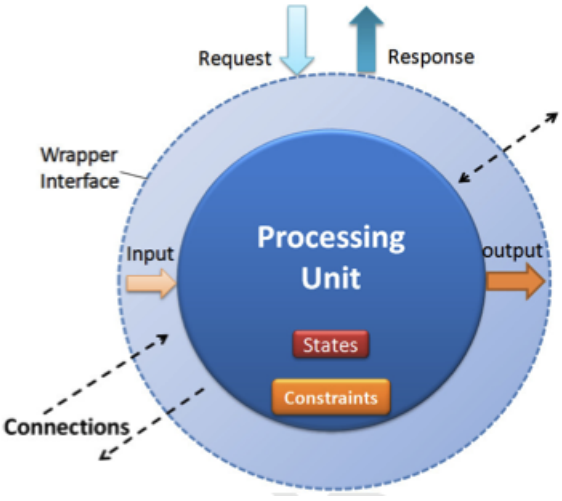


Figura 3.1: Diagrama de los elementos constitutivos de una máquina social, según el enfoque de Meira et. al. [*Nota*: Extraído de (Meira et al., 2010, Fig. 1)]

Las Relaciones $< Rel >$ de una máquina social comprenden las conexiones que establece con otras. Su Interfaz Covertora $< WI >$ es la capa que separa lo externo de lo interno, es decir, los requerimientos de otras máquinas sociales llegan a esta interfaz, y desde esta interfaz se envían las respuestas. La interfaz considera distintos tipos de interacción. Como ejemplo Meira et al. presentan la API de Twitter y el sistema de correos electrónicos de Facebook. Los requerimientos $< Req >$ pueden ser de dos tipos, requerimientos de las funcionalidades

propias de una máquina social, o bien, requerimiento de «meta-información» sobre estas funcionalidades. Meira et al. dan como ejemplos de este tipo de último tipo de requerimientos el consultar información sobre el tiempo y costo del uso de servicios, el número y tipo de parámetros correctos, y posibles restricciones, entre otros. Las respuestas $< Resp >$ pueden ser a su vez respuestas a requerimientos funcionales, respuestas de meta-información, y también respuestas de notificación, donde una máquina social alerta a las máquinas conectadas sobre su condición interna; por ejemplo, en casos de errores en tiempo de ejecución, violación de restricciones, entre otros. El Estado $< S >$ depende de cada máquina, y la información que deba almacenar para el cumplimiento de sus funcionalidades. Las Restricciones $< Const >$ son las consideraciones no funcionales que deben tenerse como reglas a la hora de permitir la conexión con otras máquinas. Meira et al. dan como ejemplo el máximo número de accesos concurrentes, protocolos de autorización, número de requerimientos por hora, entre otros. Finalmente, traspasando la barrera de la Interfaz Covertora, se encuentra el Input $< I >$, que corresponde a los datos que serán pasados como parámetro a las funciones implementadas en la Unidad de Procesamiento $< P >$, y que retornara un —posiblemente nulo— Output $< O >$.

[*Finaliza la revisión*]

3.3. Las máquinas sociales y la Adquisición de Conocimiento

Otro enfoque importante que ha recibido la noción de máquinas sociales proviene del campo del Knwolegde Acquisition (KA), o Adquisición de Conocimiento. En 2013, Shadbolt publicó un artículo titulado *«Knowledge Acquisition and the Rise of Social Machines»*, donde examinó este asunto. Lo revisamos a continuación. [*Fuente de la siguiente revisión*: (Shadbolt, 2013, pp. 201–204)]

Shadbolt hace un análisis del desarrollo del KA, dando cuenta del gran avance que el área había alcanzado hacia el inicio de los 1990s. Señala que se desarrollaban técnicas y métodos cada vez más eficiente y eficaces. Los investigadores trabajaban con la seguridad de que el área de KA proporcionaba métodos poderosos para entender el razonamiento de resolución de problemas, caracterizar estereotipos de patrones de razonamiento, e implementar estos patrones en sistemas en tiempo de ejecución. Shadbolt comenta las grandes expectativas que se tenían en el área, y relata cómo se comenzaron a construir librerías de métodos del KA, las que se ponían a prueba en «carreras por el *insight*», que buscaba aplicar nuevas formas de resolver algunos problemas clásicos, como asignación de habitaciones y diseño de ascensores. Shadbolt señala: «Nosotros creíamos confiadamente que en cualquier momento cerca del cambio de milenio el mundo de la ingeniería de software alcanzaría los límites del modelo orientado a objetos y abrazaría nuestras bibliotecas de resolución de problemas» (Ib., p. 201).

Shadbolt analiza cómo la introducción de la Web transformó la directrices del KA. La posibilidad de reunir humanos con recursos de conocimiento distribuidos a gran escala mediante la Web era precisamente una de las grandes oportunidades para la adquisición de

conocimiento, y extraer la información que residía en la Web se convirtió en uno de los principales desafíos. Shadbolt revisa los principales retos relacionados al conocimiento en la Web. Menciona la Web Semántica, que busca proveer contenido en la Web con una semántica entendible por máquinas. Y menciona también a las máquinas sociales, elemento que describe como esencial en el contexto del conocimiento a escala Web.

Shadbolt presenta como ejemplo emblemático de máquinas sociales a Ushahidi[1], una aplicación que permitió registrar lugares donde se necesitaba ayuda o donde habían actos de violencia en Kenia, luego de las elecciones del 27 de Diciembre de 2007. Las personas podían enviar sus reportes de lo que sabían vía Web o SMS, para que la comunidad de usuarios pudiera evitar los lugares donde se estaban produciendo motines y asesinatos, y también pudiera asistir a personas que lo necesitaban. Desde el punto de vista de Shadbolt, «[las] máquinas sociales son sistemas de adquisición de conocimiento a escala y máquinas que están socialmente contextualizadas» (Ib., p. 203). Según el autor, «en el corazón de las máquinas sociales está la computación social» (*Loc. cit.*). Shadbolt señala que el «programa» en una computación social generalmente parte con una especificación incompleta, pero esa completitud es un problema social más que uno matemático. «A gran escala», menciona, «la actividad basada en la Web tal como en Wikipedia o GalaxyZoo nunca converge en un ouput correcto, sino que corre continuamente e indefinidamente mientras los participantes vienen y van y el sistema crece» (Ib., p. 204).

El autor también realiza una interesante clasificación de los distintos sistemas computacionales existentes, ubicándolos en un plano de *complejidad de computación y de datos versus complejidad social*. Aquí un eje vertical mide el grado de complejidad del programa computacional y los datos necesaria para resolver el problema, y un eje horizontal mide el grado de dependencia de la solución en la interacción social a gran escala. Así, Shadbolt ubica en el plano distintos sistemas reflejando la naturaleza diferente de cada uno. Por ejemplo, el control del tráfico aéreo o la modelación del clima son actividades complejas en computación y datos pero con poca participación social —se ubican arriba a la izquierda—, mientras que el *crowdsourcing*, los sistemas de co-creación como Wikipedia y las redes sociales son comparativamente poco sofisticados pero con una alta participación social —se ubican abajo a la derecha—.

[*Finaliza la revisión*]

3.4. Un framework para clasificar máquinas sociales

En 2013, Shadbolt et al. desarrollaron un marco clasificatorio para máquinas sociales, con la finalidad de encontrar los principales constructos que identificasen a las máquinas sociales y que permitieran compararlas. Resulta muy importante revisar el trabajo de estos autores debido a que integra muchos de los conceptos revisados hasta este momento, y subraya los principales elementos involucrados en la estructura y organización de las máquinas sociales. Examinamos esto a continuación. [*Fuente de la siguiente revisión*: (Shadbolt et al., 2013, pp. 905–912)]

[1] `http://www.ushahidi.com/`

Shadbolt et al. apuntan que la noción de máquina social desdibuja la línea que separa las partes humanas y digitales de los sistemas Web, y en cambio dibuja otra pero ahora alrededor de ambas partes, permitiendo comprender cómo se entretejen los aspectos sociales y computacionales en una sola unidad. Shadbold et al. señalan: «Esencialmente las máquinas sociales pueden ser caracterizadas como ensamblajes de servicios ejecutados manualmente y dirigidos por máquinas (como en 'automatizado'), y por la interacción de tales servicios» (Ib., p. 905).

Shadbolt et al. examinan el nexo que existe entre la noción de máquinas sociales y una serie de conceptos relacionados. Los más importantes de aquellos ya fueron revisados en la sección anterior. Las nociones que mencionan los autores incluyen la de Web 2.0, *social computing*, prosumo, *wisdom of crowds*, inteligencia colectiva, *crowdsourcing*, innovación abierta y computación humana. Según Shadbolt et al., el concepto de máquinas sociales se intersecta con todas estas nociones relacionadas. Para los autores, el nacimiento de la Web 2.0 y el surgimiento del prosumo explican el contexto actual en que se expresan la noción de máquinas sociales e ideas afines. La inteligencia colectiva y el *wisdom of crowds* dan cuenta del potencial intelectual que colectivamente tienen las personas; aunque desde el punto de vista de Shadbolt et al., esto explica sólo una parte de las máquinas sociales:

> En una máquina social, la inteligencia humana y computacional confluye para alcanzar un propósito dado. Mientras la sabiduría de masas y la inteligencia colectiva ponen su foco en identificar las situaciones en que grupos de personas se desempeñan mejor que los individuos, las máquinas sociales están interesadas en el estudio y realización de sistemas híbridos en los cuales los dos tipos de componentes coexisten. Como tal, los *insights* teóricos y empíricos de estas dos áreas relacionadas son útiles para entender la dinámica de las estructuras sociales subyacentes a una máquina social, pero definitivamente no son el único ingrediente necesario para construir sistemas operacionales. (Ib., p. 906)

Shadbolt et al. continúan examinando la innovación abierta, el *crowdsourcing* y la computación humana como ideas fuertemente relacionadas a la de máquinas sociales. El uso de una fuerza de trabajo distribuida a través de la Web y los desafíos de agregación de contribuciones en una solución son, entre otros, aspectos importantes en la noción de máquinas sociales. Sin embargo, señalan los autores, la tecnología en estos casos está dispuesta para asistir objetivos específicos de la innovación abierta, el *crowdsourcing* y la computación humana. La noción de máquinas sociales, en cambio, también se interesa por medios tecnológicos para desarrollar herramientas que solucionen desafíos computacionales de naturaleza computacional; es decir, las partes humanas y computacional en las máquinas sociales son igual de importantes. Un razonamiento análogo llevan a cabo Shadbolt et al. respecto a la noción de *social computing*:

> *Social computing* pone un mayor énfasis en las capacidades de administración de la información de grupos y comunidades, y menos en la forma en que estas capacidades emergen como un esfuerzo conjunto. Esta distinción es aun mayor en el caso de máquinas sociales, que considera los componentes sociales y técnicos como socios iguales y necesarios, y estudia las formas en que podrían ser mejor combinados para vencer los desafíos de los futuros sistemas socio-técnicos. (Ib., p. 907)

El marco de clasificación que desarrollan Shadbolt et al. corresponde a una serie de cons-

tructos entendidos como características que permiten distinguir entre distintas máquinas sociales. Al usar el marco de clasificación sobre una máquina social, se evalúa la presencia o ausencia de cada característica con un puntaje de 1 a 5, donde 1 significa ausencia o mínima intensidad del constructo y 5 significa presencia o máxima intensidad de éste. Al usar el marco sobre un conjunto de máquinas, los resultados pueden ordenarse según un dendrograma, lo que permite visualizar mejor las similitudes y diferencias entre las máquinas sociales. La Figura 3.2 es una reproducción de aquella que Shadbolt et al. utilizaron para exhibir los constructos de su marco de clasificación.

Tasks, purpose and context of participation
Activities involve creative production of content
Activities involve subjective appraisal of content
Activities involve solving (a definable) computation task or set of problems
Tasks are domain-specific
The machine owner derives benefit from participation
Activities and tasks are pre-defined or participant-defined
Variation in types of contributions and tasks
Participants' participate to fulfil needs of a role
work-related/professional role
home/family related
social role
leisure/entertainment role
Participation is done via:
mobile devices
Web browsers
apps
sensors/sensing (location sensing and wearable devices)
Participation is done in a mobile context
Physical location is relevant to the service
Participants and roles
Generality of audience
Participant autonomy
Participant anonymity
Extent of hierarchical organisation of roles
Clear separation of roles among participants
Motivation and incentives
Participants are intrinsically motivated:
to gain/share knowledge
to "get something done"
to "be for fun/entertainment"
to "be social"
for the benefit of a specific group of people who need help
for the benefit of society as a whole
Participation is motivated by extrinsic reward (payment, status)

Figura 3.2: Constructos extraídos por Shadbolt et al. para la clasificación de máquinas sociales. [*Nota*: Extraído de (Shadbolt et al., 2013, Tabla 1)]

En términos generales, los constructos se ordenan por tres grandes áreas: *Tareas, propósitos y contexto de la participación*; *Participantes y roles*; y *Motivación e incentivos*. Esto equivale, en términos más reducidos, a decir que se ordenan por contribuciones, participantes y motivaciones. La primera de estas áreas reúne calificaciones sobre la creatividad de la contribución y el grado de subjetividad. Por ejemplo, si el constructo sobre creatividad obtiene un 1, entonces la actividad no requiere producir contenido creativo. Si en cambio obtiene un 5, entonces la actividad se basa en la capacidad creativa de los contribuyentes. Así mismo encontramos calificaciones sobre si las actividades requieren resolver tareas computacionales, si las tareas son de dominio específico, si el dueño del sistema se beneficia de la participa-

ción de los usuarios, si las tareas están pre-definidas o los participantes las definen, y si las contribuciones varían su tipo o no. También se incluye una serie de constructos para evaluar el contexto en que sucede la participación: laboral, familiar, social o de entretención. Así mismo se incluyen constructos para evaluar los medios por los cuales se participa, y el grado de dependencia geográfica del servicio. La segunda área está formada por constructos para distinguir las características de los participantes y su organización. Aquí se evalúa si la audiencia es general o específica, el nivel de autonomía y anonimato de los participantes, y el grado de jerarquía y separación en los roles. Finalmente, la última área contiene constructos para clasificar la motivación de los usuarios en términos de por qué participan.

Algo que es muy interesante de destacar es que usando este marco clasificatorio, Shadbolt et al. realizaron un análisis de 20 máquinas sociales llegando a muy buenos resultados en términos de medir similitudes y diferencias entre las máquinas. El dendrograma que obtuvieron se reproduce en la Figura 3.3. Por ejemplo, YouTube, Vimeo, Reddit y Digg resultaron ser similares, lo que para Shadbolt et al. representa su característica de ser sistemas donde se comparte contenido. La misma similitud se dio entre Quora y Stack Overflow, lo que los autores entienden debido a que ambos sistemas centran su actividad en responder preguntas. Shadbolt et al. terminan concluyendo que estos constructos dan una base coherente e integral para la descripción y clasificación de las máquinas sociales, así como también para la comparación directa entre ellas.

Figura 3.3: Dendrograma que Shadbolt et al. obtuvieron al comparar 20 máquinas sociales bajo su framework clasificatorio. [*Nota*: Extraído de (Shadbolt et al., 2013, Fig. 3)]

[*Finaliza la revisión*]

3.5. Una visión unificadora para las máquinas sociales

El último enfoque que revisaremos integra explícitamente las diferentes ideas y nociones que hemos estudiado bajo el concepto de máquinas sociales. Fue propuesto por Burégio, Meria

y Rosa en su artículo *«Social Machines: A Unified Paradigm to Describe Social Web-Oriented Systems»* de 2013. Burégio et al. describen el tópico de «Máquinas Sociales» como el área de investigación que se ocupa de los desafíos que se originan al «fundir» elementos sociales y computacionales en el software. Desde su punto de vista, sin embargo, aún no están claros los límites del área, y no se han podido integrar completamente los distintos enfoques. Los autores señalan:

> A pesar de ser un tópico prometedor, los conceptos detrás de Máquinas Sociales se superponen con distintos campos de investigación y, consecuentemente, han creado confusión y planteado varias interrogantes. Por ejemplo, hemos encontrado algunos investigadores que han tenido dificultades en entender los límites de este tópico de investigación y cómo este puede contribuir a sus campos de investigación. (Burégio et al., 2013, p. 885)

Burégio et al. buscan caracterizar el área como un todo. Investigan los trabajos relacionados a la noción de máquinas sociales y los reúnen bajo una visión general unificadora. De esta forma los autores integran los distintos aportes, trabajos y enfoques sobre máquinas sociales en un «esquema de clasificación común y convergente» que les permite, por un lado, identificar las principales visiones con que se investiga, y por otro, generar un cuerpo unificado de la noción de máquinas sociales. Más aún, los autores postulan que el tópico de máquinas sociales representa un «paradigma prometedor» para describir las entidades en la Web.

Burégio et al. hacen converger los distintos trabajos en el área de máquinas sociales en torno a tres grandes visiones: *Software Social*, *Personas como Unidades Computacionales*, y *Software como Entidades Sociables*. Estas visiones en conjunto caracterizan el «paradigma» de máquinas sociales. Las revisamos a continuación. [*Fuente de la siguiente revisión*: (Burégio et al., 2013, pp. 886–890)]

Software Social. Para Burégio et al., esta visión considera las tecnologías que permiten interactuar, colaborar y compartir con otros, junto al software que se genera para ello. Están las máquinas sociales tempranas, que son «una generación inicial de software social basado en la Web (colectivamente llamado "Web 2.0" que consiste de blogs, redes sociales, sitios para compartir videos, etc)» (Ib., p. 886). Aquí se encuentra Facebook y Tweeter, por ejemplo. También están las plataformas Open API que desarrollan los sitios de redes sociales —e.g. Tweeter API—, y que permite a otros desarrolladores interactuar y acceder a la información de redes sociales para montar aplicaciones sobre ellas. Están también los denominados *mashups*, nuevas aplicaciones que son el resultado de integrar servicios de aplicaciones ya existentes —e.g. el caso de iTransantiago, visto como la suma de GoogleMaps y los datos del Transantiago—. Y también están los sistemas que recogen, analizan y utilizan los datos sociales para inferir preferencias, proveerlas a otras aplicaciones, o incluso realizar acciones sobre objetos reales. Este último es el caso de iStrategyLabs[2], iniciativa que lleva a cabo actividades reales físicas mediante el procesamiento de información en la forma de, por ejemplo, Tweets o Likes de Facebook.

Personas como Unidades Computacionales. En esta visión encontramos como pilares fundamentales las nociones de computación humana, *microtasking*, y *crowdsourcing*, que ya hemos examinado con anterioridad. Estos conceptos son efectivamente parte de la

[2] `http://istrategylabs.com/`

noción de máquinas sociales según el enfoque integrador de Bugério et al., y caracterizan a toda una visión agrupadora completa. Aquí se menciona como ejemplo a reCAPTCHA, GWAPs, y Mechanical Turk, entre otros. Además esta visión reúne el enfoque de máquinas sociales cómo sistemas de adquisición de conocimiento (Shadbolt, 2012), lo que para Burégio et al. le vale a Wikipedia y Ushahidi un puesto en este grupo.

Software como entidades sociables. Esta visión integra los trabajos que avanzan en la dirección de comprender las relaciones que pueden generarse entre máquinas sociales. Aquí se consideran estudios que investigan «interacciones sociales» entre servicios Web, así como las propuestas que analizan posibles comunidades o redes sociales de estos. También se consideran las investigaciones sobre descripciones dinámicas de servicios Web que permitan que estos reconozcan en sus pares relaciones de colaboración, competencia o sustitución, dándoles el carácter de entidades sociables que interactúan entre sí.

[*Finaliza la revisión*]

Parte II

Formalizando la noción de máquinas sociales

En la segunda parte de esta memoria se realiza la formalización de la noción de máquinas sociales. Primero se ofrece una revisión de las nociones formales básicas necesarias para la comprensión del modelo que se presentará y luego se explica en detalle el modelo formal de máquinas sociales elaborado durante el trabajo de título.

Capítulo 4

Preliminares

En este capítulo preliminar revisaremos las nociones formales elementales para comprender el modelo de máquinas sociales que se presentará posteriormente. En la primera sección de este capítulo se introducen los conceptos matemáticos básicos involucrados en el modelo. En la segunda sección se definen los conceptos de máquina de Turing y Turing computabilidad. En la tercera sección se estudia el concepto de *stream* y finalmente en la cuarta y última sección se presenta el formalismo del π-cálculo, sobre el cual descansa parte importante del modelo de máquinas sociales que se presentará en el próximo capítulo.

4.1. Nociones básicas

En esta sección introduciremos brevemente parte de las nociones matemáticas básicas que se ocuparán en el modelo formal de máquinas sociales. También estableceremos las notaciones y convenciones básicas que se utilizarán más adelante. Como es usual se recomienda al lector dar una lectura rápida a esta sección u omitirla y volver aquí en caso que lo necesite.

Usaremos los símbolos $\wedge$, $\vee$, $\neg$, $\Rightarrow$ y $\Leftrightarrow$ para los conectivos lógicos de conjunción, disyunción, negación, si ... entonces y si y solo si, como es usual. También los símbolos $\forall$ y $\exists$ denotarán los cuantificadores para todo y existe. Se advierte al lector que los símbolos $\rightarrow$, $\vdash$ y $\vDash$ —que también son utilizados en lógica— serán reservados para otros conceptos con significado distinto. También $\notin$ denotará la negación de $\in$, $\nvdash$ la negación de $\vdash$ y así en adelante.

Un conjunto será una colección de objetos. Los símbolos $\mathbb{N}$ y $\mathbb{P}$ denotarán el conjunto de los números naturales y el conjunto de los números positivos, respectivamente. El conjunto vacío lo denotaremos por $\emptyset$, y si A es un conjunto entonces $\mathscr{P}(A)$ será el conjunto de todos los subconjuntos de A, que llamaremos conjunto potencia. Dados dos conjuntos A y B, denotaremos su unión, intersección y diferencia por $A \cup B$, $A \cap B$ y $A - B$, como es usual. También si X es una familia de conjuntos entonces la unión de X será el conjunto:

$$\bigcup X = \{a : \exists A \in X,\ a \in A\} \tag{4.1}$$

Similarmente:

$$\bigcap X = \{a : \forall A \in X,\ a \in A\} \tag{4.2}$$

Dados dos objetos a y b, se define el par ordenado (a, b) como $(a, b) = \{\{a\}, \{a, b\}\}$. A su vez se define un trío ordenado como $(a, b, c) = ((a, b), c)$, y en general se define inductivamente una $(n+1)$-tupla ordenada como:

$$(x_1, \ldots, x_{n+1}) = ((x_1, \ldots, x_n), x_{n+1}) \tag{4.3}$$

Donde $(x_1, \ldots, x_n)$ denota una n-tupla y donde el caso base es el par ordenado. Llamaremos a n el largo de la tupla ordenada, y definimos por completitud la tupla de largo 1 como $(x) = x$. Dada una n-tupla $x = (x_1, \ldots, x_n)$ llamamos a cada objeto de la forma x_i la i-ésima componente de x. Una secuencia s es simplemente una n-tupla para algún $n > 0$. Algunas veces omitiremos la notación de n-tupla ordenada al hablar de secuencias y escribiremos simplemente $a_1, \ldots, a_n$ en vez de $(a_1, \ldots, a_n)$. También privilegiaremos el término tupla por sobre n-tupla ordenada cuando esto resulte más conveniente. Una definición importante respecto a tuplas la damos a continuación.

Definición 4.1 *Sean $a = (a_1, \ldots, a_n)$ y $b = (b_1, \ldots, b_m)$ dos tuplas cualesquiera. Se define el operador de* concatenación *de tuplas, símbolo $\langle\ ,\ \rangle$, como $\langle a, b\rangle = (a_1, \ldots, a_n, b_1, \ldots, b_m)$*

Dados dos conjuntos A y B, definimos $A \times B = \{(a, b) : a \in A \wedge b \in B\}$. Extendemos la noción a n conjuntos $A_1, \ldots, A_n$ definiendo $A_1 \times \ldots \times A_n$ como el conjunto de todas las n-tuplas $(a_1, \ldots, a_n)$ tales que $a_i \in A_i$. También usaremos la notación $\prod_{i=1}^{n} A_i$ cuando resulte conveniente, y definimos $A^n = \prod_{i=1}^{n} A$. Una relación n-aria entre los conjuntos $A_1, \ldots, A_n$ es un conjunto $\mathcal{R}$ tal que $\mathcal{R} \subseteq A_1 \times \ldots \times A_n$. Cuando $\mathcal{R}$ sea una relación binaria, escribiremos también $a\mathcal{R}b$ como alternativa a $(a, b) \in \mathcal{R}$. En general omitiremos la información sobre la aridad de las relaciones cuando ésta se subentienda.

Ahora corresponde definir el concepto de función. Es importante dejar claras las convenciones que tomaremos respecto al uso de esta noción, ya que trabajaremos en el modelo con funciones totales y parciales. Establecemos la siguiente definición:

Definición 4.2 *Dados dos conjuntos A y B, una función f de A en B es un conjunto $f \subseteq A \times B$ que satisface:*

$$\forall a \in A \quad \forall b, c \in B \quad (a, b) \in f \wedge (a, c) \in f \Rightarrow b = c \tag{4.4}$$

$$\forall a \in A \quad \exists b \in B \quad (a, b) \in f \tag{4.5}$$

Si f es una función de A en B anotamos $f : A \to B$ y decimos también que f es una función total. Si $f \subseteq A \times B$ es tal que satisface (4.4) *pero no necesariamente* (4.5)*, entonces decimos que f es una función parcial de A en B y anotamos $f : A \nrightarrow B$. Si $(a, b) \in f$ anotamos $f(a) = b$ y decimos que f está definida en $a \in A$. También, para toda función f total o parcial se definen los conjuntos dominio e imagen de f como sigue:*

$$Dom(f) = \{a \in A : \exists b \in B \quad f(a) = b\} \tag{4.6}$$

$$Img(f) = \{b \in B : \exists a \in A \quad f(a) = b\} \tag{4.7}$$

$$\Im \tag{4.8}$$

Por último, el espacio de todas las funciones totales de A en B y el espacio de todas las funciones parciales de A en B se definen como sigue:

$$A \to B = \{f \subseteq A \times B \ : \quad f : A \to B\} \tag{4.9}$$

$$A \nrightarrow B = \{f \subseteq A \times B \ : \quad f : A \nrightarrow B\} \tag{4.10}$$

De esta forma, a menos que se indique explícitamente lo contrario, por función nos referimos a función total.

Dada f una función de A en B total o parcial, si para todo $a, a' \in Dom(f)$ se tiene que $a \neq a' \Rightarrow f(a) \neq f(a')$ entonces diremos que f es inyectiva. Si $Img(f) = B$ entonces f es sobreyectiva. Y si f es inyectiva y sobreyectiva entonces diremos que es biyectiva. Sea $f : A \to B$ y $g : C \to D$ tales que $Img(f) = C$. Entonces se define la composición de f con g como $g \circ f : A \to D$ tal que $(g \circ f)(a) = g(f(a))$. También se definen los operadores de proyección de la siguiente forma:

Definición 4.3 *Sea $A \subseteq A_1 \times \ldots \times A_n$. Entonces para cada $i = 1, \ldots, n$, se define el operador de la proyección i-ésima, notación π_i, como:*

$$\pi_i : A \to A_i, \quad \pi_i(a_1, \ldots, a_n) = a_i \tag{4.11}$$

Una definición que será útil más adelante es la siguiente:

Definición 4.4 *Sea f una función de A en B total o parcial, tal que $f(a) = b$ para algún $a \in A$ y $b \in B$. Entonces, dado $c \in B$, denotaremos por $f[c/a]$ la función idéntica a f en todos sus puntos, excepto en a donde remplaza la imagen de a por c. De esta forma, $f[c/a]$ se define como:*

$$f[c/a] = (f - \{(a, b)\}) \ \cup \ \{(a, c)\} \tag{4.12}$$

Otro concepto que usaremos bastante es el de indización. Primero, definimos para cada número $n > 0$ el conjunto I_n de los primeros n positivos como:

$$I_n = \{ \ i \in \mathbb{N} : 1 \leq i \leq n\} \tag{4.13}$$

Luego, sea A un conjunto finito tal que $n = |A|$. Una indización de A es una función biyectiva definida como index : $I_n \to A$. Para cada $i \in I_n$, denotamos el objeto index(i) por a_i, y llamamos a i el índice del objeto a_i. Dado cualquier conjunto A, toda indización de A induce un ordenamiento en A, a saber, $a_i < a_j$ si $i < j$. Finalmente, cuando digamos «sea I un conjunto de índices de A» estaremos asumiendo implícitamente la existencia de una biyección fija entre I y A.

Ahora corresponde introducir las nociones de alfabeto, lenguaje y operaciones entre lenguajes. Llamaremos alfabeto a cualquier conjunto finito no vacío. A los elementos de un alfabeto los llamaremos símbolos o caracteres. Así, las letras a, b, z son ejemplos de símbolos, como también #, $\sqcup$, $\triangleright$, entre otros. Por lo general, emplearemos las letras griegas mayúsculas Σ, Γ, y Λ para denotar alfabetos. Un string sobre un alfabeto Σ es una secuencia finita de símbolos en Σ, posiblemente vacía, es decir, es un objeto de la forma:

$$w \ \in \ \Sigma^* = \bigcup_{k \in \mathbb{N}} \Sigma^k \tag{4.14}$$

Donde $\Sigma^1 = \Sigma$, y $\Sigma^k = \Sigma \times \Sigma^{k-1}$. El conjunto Σ^0 es un conjunto especial que contiene solo un string, el string que no contiene ningún símbolo y que es llamado string vacío y es denotado por ε. También usamos el término palabra y cadena para referirnos a un string. Al escribir strings omitimos la notación de tupla y simplemente escribimos las componentes una seguida de otra. Así, el string $(1, 0, 1, 1, 0)$ lo escribimos simplemente como 10110. También, anotamos la i-ésima componente de un string w como $w[i]$. El largo de un string w es su largo como n-tupla ordenada; lo anotamos como $|w| = n$. El largo del string vacío ε es 0. Si u y v son strings, entonces uv representa la concatenación de u y v.

Un lenguaje L sobre un alfabeto Σ será un conjunto de strings sobre Σ, es decir, $L \subseteq \Sigma^*$. Junto con las operaciones básicas de conjuntos, los lenguajes se pueden combinar mediante las operaciones que definimos a continuación:

Definición 4.5 *Sean L_1 y L_2 lenguajes sobre Σ. Se definen las siguientes operaciones sobre lenguajes:*

1. *Concatenación:* $L_1L_2 = \{xy : x \in L_1 \wedge y \in L_2\}$
2. *Potencia:* $L_1^0 = \{\varepsilon\}, L_1^k = L_1L_1^{k-1}$
3. *Clausura de Kleene:* $L_1^* = \bigcup_{k\in\mathbb{N}} L_1^k$

4.2. Máquinas de Turing

Revisamos a continuación la noción de máquinas de Turing. Hacemos esto por dos razones principales. La primera es que permitirá al lector acceder fácilmente a las definiciones del formalismo de máquinas de Turing en caso que desee ir comparándolo en el transcurso de la lectura con el modelo de máquinas sociales que se presentará. La segunda razón es que algunos resultados del modelo que se expondrá en el próximo capítulo descansan explícitamente sobre las nociones formales de máquinas de Turing y Turing computabilidad, y por ello, es necesario establecer con claridad qué es lo que vamos a entender por estos conceptos, debido a que hay más de una formulación posible. Introducimos estas nociones a continuación.

Definición 4.6 *Una* Máquina de Turing *es una tupla $T = (K, \Sigma, \delta, s, H)$, donde*

- *K es un conjunto finito de* estados*;*
- *Σ es un alfabeto finito que contiene el símbolo blanco # pero no los símbolos L ni R;*
- *$s \in K$ es el* estado inicial*;*
- *$H \subseteq K$ es el conjunto de* estados de parada*;*
- *$\delta : (K - H) \times \Sigma \to K \times (\Sigma \cup \{L, R\})$ es la* función de transición.

El enfoque que seguimos en esta sección está inspirado en el texto clásico de teoría de la

computación *«Elements of the Theory of Computation»* de Lewis y Papadimitriou, de 1981. La definición de las nociones de configuración, lleva en un paso y computación se establecen como es usual de la siguiente manera:

Definición 4.7 *Una* configuración *de una máquina de Turing* $T = (K, \Sigma, \delta, s, H)$ *es un elemento del conjunto*

$$\mathcal{C}_T = K \times \Sigma^* \times \Sigma \times (\Sigma^*(\Sigma - \{\#\}) \cup \varepsilon) \tag{4.15}$$

Una configuración de la forma (q, u, a, v) *se escribirá también* $(q, u\underline{a}v)$*. Además, si una configuración* (q, u, a, v) *es tal que* $q \in H$*, entonces diremos que es una configuración de parada.*

Definición 4.8 *Sea* $T = (K, \Sigma, \delta, s, H)$ *una máquina de Turing. Se define la relación lleva en un paso,* $\vdash_T \subseteq \mathcal{C}_T \times \mathcal{C}_T$*, de la siguiente forma:*

$$(q, u, a, v) \vdash_T (q', u', a', v')$$

si y solo si, para algún $b \in \Sigma \cup \{L, R\}$*,* $\delta(q, a) = (q', b)$ *y se cumple una de las siguientes condiciones:*

1. $b \in \Sigma$*,* $u = u'$*,* $v = v'$*, y* $a' = b$*;*
2. $b = L$*,* $u = u'a'$ *y se cumple ya sea:*
   (a) $v' = av$*, si* $a \neq \# \vee v \neq \varepsilon$*, o bien*
   (b) $v' = \varepsilon$*, si* $a = \# \wedge v = \varepsilon$*;*
3. $b = R$*,* $u' = ua$*, y se cumple ya sea:*
   (a) $v = a'v'$*, o bien*
   (b) $v = v' = \varepsilon \wedge a' = \#$*.*

Definición 4.9 *Dada una máquina de Turing* $T = (K, \Sigma, \delta, s, H)$*, el símbolo* $\vdash_T^*$ *denota la clausura refleja y transitiva de* $\vdash_T$*; diremos que una configuración* C_1 lleva en cero o más pasos *a una configuración* C_2 *si* $C_1 \vdash_T^* C_2$*. Una* computación *de* T *es una secuencia de configuraciones* $C_0, C_1, \ldots, C_n$ *para algún* $n \geq 0$ *tal que:*

$$C_0 \vdash_T C_1 \vdash_T \ldots \vdash_T C_n.$$

Ahora que hemos definido las nociones de máquina de Turing, configuración, lleva en un paso y computación, podemos establecer las nociones de Turing computabilidad que serán utilizadas al trabajar en el modelo de máquinas sociales.

Definición 4.10 *Sea* Σ_0 *un alfabeto que no contiene el símbolo blanco* $\#$*. Sea* f *una función de* Σ_0^* *en* Σ_0^* *total o parcial. Sea* $T = (K, \Sigma, \delta, s, H)$ *una máquina de Turing. Definimos las siguientes nociones de computación:*

1. *Si f es una función total, diremos que T* computa *f si y solo si $\Sigma_0 \in \Sigma$ y para cada $w \in \Sigma_0^*$ se cumple:*

$$f(w) = u \Rightarrow (s, \#w\underline{\#}) \vdash_T^* (h, \#u\underline{\#}), h \in H \tag{4.16}$$

   Si existe tal T diremos que f (total) es Turing computable.

2. *Si f es una función parcial, diremos que T* computa *f si y solo si $\Sigma_0 \in \Sigma$ y para cada $w \in \Sigma_0^*$ se cumple:*

   si f está definida en w, se satisface (4.16)
   si f no está definida en $w \Rightarrow$ T diverge con input w

   Si existe tal T diremos que f (parcial) es Turing computable.

3. *Si f es una función parcial, diremos que T* computa parcialmente *f si y solo si $\Sigma_0 \in \Sigma$ y para cada $w \in \Sigma_0^*$ se cumple:*

   si f está definida en w, se satisface (4.16)
   si f no está definida en $w \Rightarrow$ T puede o no divergir con input w

   Si existe tal T diremos que f es parcialmente Turing computable.

Las definiciones anteriores se extienden al caso de funciones con múltiples argumentos de entrada y salida reemplazando la ecuación (4.16) *por:*

$$\begin{gathered} f(w_1, \ldots, w_n) = (u_1, \ldots, u_m) \Rightarrow \\ (s, \#w_1\# \ldots \#w_n\underline{\#}) \vdash_T^* (h, \#u_1\# \ldots \#u_m\underline{\#}), h \in H \end{gathered} \tag{4.17}$$

4.3. Streams

A continuación introducimos el concepto de streams. Esta noción jugará un rol importante dentro del modelo formal de máquinas sociales; definiciones y resultados significativos descansan sobre este concepto. Por ello es de gran relevancia tener una idea clara respecto de qué son estos objetos matemáticos y cómo se trabaja con ellos. Lamentablemente sucede que formular una definición matemática precisa del concepto de streams está fuera de los alcances de esta memoria; por esta razón, procederemos dando una explicación intuitiva sobre streams, y luego entregaremos pinceladas iniciales al lector sobre cómo se construyen estos objetos matemáticamente, para posteriormente remitir a quién lo desee a la bibliografía pertinente.

Comenzamos dando una introducción intuitiva del concepto de stream. Sea A un conjunto. Un stream sobre A es un par ordenado de la forma $s = (a, s')$ tal que $a \in A$ y s' es otro stream sobre A. Esto es, un stream es un elemento de A seguido de otro stream. Por ejemplo, sea $A = \{0, 1, 2\}$ y consideremos el caso más simple, el stream constante c_0 tal que $c_0 = (0, c_0)$. Entonces tenemos que:

$$c_0 = (0, c_0) = (0, (0, c_0)) = (0, (0, (0, c_0))) \tag{4.18}$$

En general, cuando trabajemos con streams usaremos los paréntesis angulares $\langle\ ,\ \rangle$ de forma intercambiable con los paréntesis redondos (,). Continuando con el ejemplo, otro stream podría ser el siguiente:

$$s = \langle 1, s' \rangle = \langle 1, \langle 2, s'' \rangle \rangle = \langle 1, \langle 2, \langle 0, s''' \rangle \rangle \rangle \tag{4.19}$$

Donde s''' es un stream sobre A. Cuando escribamos streams, nos daremos permiso de omitir componentes que sean streams si lo que nos interesa es conocer los elementos de A. Así la ecuación (4.19) quedaría:

$$s = \langle 1, \ldots \rangle = \langle 1, \langle 2, \ldots \rangle \rangle = \langle 1, \langle 2, \langle 0, \ldots \rangle \rangle \rangle \tag{4.20}$$

También podemos utilizar funciones para definir streams. Por ejemplo, sea f tal que:

$$f(n) = \langle n, f(n+1) \rangle$$

Entonces, para cualquier n, $f(n)$ es un stream. Por ejemplo:

$$f(0) = \langle 0, \langle 1, \langle 2, \ldots \rangle \rangle \rangle$$

Introducimos más formalmente el concepto de streams mediante la siguiente definición:

Definición 4.11 *Sea A un conjunto cualquiera. El conjunto A^∞ de streams sobre A se define como el conjunto más grande que satisface la siguiente ecuación:*

$$A^\infty = A \times A^\infty \tag{4.21}$$

Donde la relación de orden corresponde a la inclusión de conjuntos.

A continuación establecemos dos definiciones que serán muy útiles al desarrollar el modelo de máquinas sociales, y enunciamos, sin demostración, un principio fundamental para trabajar con streams.

Observación 4.12 En la siguiente definición introducimos una versión acotada de los operadores de proyección que ya habían sido definidos en la primera sección, Definición 4.3.

Definición 4.13 *Sea A^∞ el conjunto de streams sobre un conjunto A. Definimos los operadores de proyección π_1 y π_2 como:*

$$\pi_1(\langle a, s' \rangle) = a \tag{4.22}$$
$$\pi_2(\langle a, s' \rangle) = s' \tag{4.23}$$

Además definimos la n-ésima composición de π_2 como:

$$\pi_2^0(s) = s, \qquad \pi_2^1(s) = \pi_2(s), \qquad \pi_2^{n+1}(s) = \pi_2^n(\pi_2(s)) \tag{4.24}$$

Definición 4.14 *Sea A^∞ el conjunto de streams sobre un conjunto A. Sea $\sigma = \langle a', \sigma' \rangle \in A^\infty$. Diremos que $a \in A$ es una componente de σ y anotaremos $a \prec \sigma$ si y solo si se cumple que $a = a'$ o bien a es una componente de σ'. Equivalentemente, $a \prec \sigma$ si y solo si $\exists n \in \mathbb{N}$ tal que $a = \pi_1(\pi_2^n(s))$.*

Proposición 4.15 (Principio de Coinducción para Streams) *Sea A^∞ el conjunto de streams sobre un conjunto A. Para demostrar que un conjunto Z es tal que $Z \subseteq A^\infty$, basta mostrar que $Z \subseteq A \times Z$.*

Como el lector ya se habrá dado cuenta, la noción de streams se destaca por su carácter circular, y en su definición subyace una relación que no está bien fundada. En efecto, los streams pueden verse como objetos matemáticos que aparecen dentro de una teoría axiomática de conjuntos que es una variante de la axiomatización de Zermelo-Fraenkel (ZF), y que busca poder modelar fenómenos circulares que ZF no puede. En este sentido, un stream es una solución a un sistema de ecuaciones de la forma:

$$\begin{aligned} x_1 &= \langle a_1, x_2 \rangle \\ x_2 &= \langle a_2, x_3 \rangle \\ &\vdots \end{aligned}$$

Estas ecuaciones son una generalización de un caso más básico de ecuaciones denominado sistemas de ecuaciones de conjuntos. Por ejemplo, la siguiente ecuación describe un sistema de ecuaciones de conjuntos de una indeterminada:

$$x = \{x\} \tag{4.25}$$

Así, si Ω es una solución de este sistema entonces:

$$\Omega = \{\Omega\} = \{\{\Omega\}\} = \{\{\{\Omega\}\}\} \tag{4.26}$$

Una relación binaria $\mathcal{R}$ en un conjunto S se dice bien fundada si no existe una sucesión infinita $b_0, b_1, b_2 \ldots$ de elementos de S tal que $b_{n+1} \mathcal{R} b_n$ para cada $n = 0, 1, 2, \ldots$; en ZF, el axioma de fundación establece que para cada conjunto A, la relación binaria $\in$ es bien fundada en los elementos de A. Claramente este no podría ser el caso del conjunto Ω, ya que $\ldots \in \Omega \in \Omega \in \Omega$[1]; de esta forma es necesario dejar de lado el axioma de fundación e incorporar en cambio otros axiomas que permitan modelar circularidades. Remitimos al lector que desee conocer una formulación clara y precisa del concepto de streams a Barwise y Moss (1996).

4.4. π-cálculo

El π-cálculo es un cálculo de procesos que sirve para describir procesos concurrentes y sus interacciones. Este formalismo que será utilizado como base para describir interacciones entre máquinas sociales y agentes, por lo que resulta indispensable introducirlo en este capítulo. Primero se ofrece una descripción intuitiva del π-cálculo para luego proceder a definirlo formalmente junto con las nociones de congruencia y reducción.

El π-cálculo consiste en un conjunto de expresiones que siguen una sintaxis específica. Estas expresiones del π-cálculo, también denominadas expresiones de procesos del π-cálculo

[1]Es decir, tomamos la sucesión infinita de elementos de Ω como $b_i = \Omega$ para todo i, y encontramos que $\in$ no es bien fundada en Ω.

o términos del π-cálculo, representan intuitivamente procesos computacionales de la vida real; el π-cálculo especifica un cierto conjunto de reglas para la construcción de expresiones nuevas a partir de otras pre-existentes, lo que representa intuitivamente las distintas formas en que pueden relacionarse los procesos. El concepto que fundamenta la dinámica del π-cálculo es el de interacción entre procesos. Dos procesos interactúan cuando intercambian mensajes a través de un canal. De hecho, la interacción a través de canales es la única forma de computación existente en el π-cálculo; la única característica observable de los procesos es su capacidad de enviar y recibir mensajes. A continuación revisaremos las nociones principales del π-cálculo.

Primero que todo tenemos el proceso inerte 0; este es un proceso que no tiene ningún comportamiento. Luego, si P es una expresión del π-cálculo que contiene el nombre x, entonces la expresión:

$$P\{z/x\} \qquad \text{(reemplazo)}$$

representa el proceso en que todas las instancias de x han sido reemplazadas por z. Este reemplazo podemos pensarlo como el acto de redefinir una variable del proceso P. Luego nos encontramos con las expresiones para enviar y recibir un nombre por un canal. Dada la expresión de proceso P, la expresión:

$$\bar{u}z.P \qquad \text{(prefijo de output)}$$

denota el proceso que está esperando enviar el nombre z por el canal de nombre u para luego activar el proceso P. Llamamos a esta operación prefijo de output, y decimos que el proceso está esperando enviar un nombre por un cierto canal. En muchos casos será útil rodear el nombre a enviar por paréntesis, por lo que también escribiremos el prefijo de output como $\bar{u}\langle z\rangle.P$. Complementariamente, si P es un término del π-cálculo, entonces el término:

$$u(x).P \qquad \text{(prefijo de input)}$$

denota el proceso que espera recibir un nombre z por el canal de nombre u, para luego activar el proceso $P\{z/x\}$. A esta operación la llamamos prefijo de input, y decimos que el proceso está escuchando por un cierto canal o que está esperando recibir un nombre por ese canal. Se debe observar que lo que se envía por los canales son nombres, y que los mismos canales están representados en las expresiones a través de sus nombres. De esta forma, por ejemplo, la expresión $\bar{z}z.\,0$ significa «envía por el canal de nombre z el nombre del mismo canal, y luego queda inerte». Luego tenemos la operación de composición paralela. Dado dos procesos, denotados por P y Q, la expresión:

$$P|Q \qquad \text{(composición)}$$

Representa el proceso formado por dos subprocesos, P y Q, que se ejecutan concurrentemente. Es gracias a la noción de concurrencia que un proceso puede interactuar con otro, como veremos más adelante. Continuando, si P es un término cualquiera, entonces la expresión:

$$(a).P \qquad \text{(restricción)}$$

señala que el nombre a es nuevo y es de uso exclusivo de P. Con esta operación, por ejemplo, se tiene que dado un proceso $P := u(x).P'$ y otro $Q := (u).\bar{u}z.Q'$, entonces P y Q están escuchando por canales distintos, y por lo tanto no se podrán comunicar. Aún cuando el nombre alfabético de los canales sea el mismo, la operación de restricción crea un nuevo nombre único

y exclusivo para el proceso en cuestión. También existe el operador de replicación. Dada una expresión de proceso P, la expresión:

$$!P \qquad \text{(replicación)}$$

representa el proceso que tiene infinitas copias del proceso denotado por P corriendo en paralelo, es decir, $P|P|P|\ldots$; este tipo de expresiones permite describir procesos que se programan para correr indefinidamente. Por último, la sumatoria de expresiones del π-cálculo representa un proceso que de forma no determinista se puede convertir en cualquiera de sus sumandos. Por ejemplo, si P y Q son dos expresiones de procesos, entonces la expresión:

$$P + Q \qquad \text{(suma)}$$

representa un proceso que puede ser o bien P o bien Q. Este tipo de expresiones es muy útil para describir situaciones que pueden ocurrir de más de una forma. Por ejemplo, si un proceso P espera un mensaje para dar lugar a un proceso Q, y P escucha por dos canales, entonces $P := c_1(x).Q + c_2(x).Q$ describe correctamente esta situación. Nótese que si hubieramos escrito $P := c_1(x).Q \mid c_2(x).Q$, entonces si P recibe un mensaje, se dispararía el proceso Q, pero aún se estaría esperando por el otro canal un nombre, lo que después lanzaría una nueva copia de Q. Por último, cabe mencionar que cada vez que el proceso inerte 0 vaya precedido de algún prefijo, entonces lo omitiremos, escribiendo $u(x)$ en vez de $u(x).\,0$, a la vez que $\bar{u}z$ en vez de $\bar{u}z.\,0$.

Ahora que ya hemos conocido la mayoría de las primitivas del π-cálculo, introducimos el concepto de interacción. En el π-cálculo se produce una interacción cuando dos procesos que corren en paralelo se comunican, al disponerse de tal forma que uno está esperando enviar un nombre por un determinado canal y el otro está escuchando por ese mismo canal. Está noción de interacción se captura formalmente en el π-cálculo a través de la relación de reducción $\rightarrow$, la que definiremos formalmente más adelante; aquí vamos a dar una idea intuitiva de este concepto. Para facilitar la exposición del tema, de ahora en adelante nos permitiremos llamar procesos a las expresiones mismas del π-cálculo, sobrentendiendo por toda la explicación anterior que, en estricto rigor, los procesos son las entidades denotadas por las expresiones. Supongamos que tenemos los siguientes procesos:

$$P := u(x).P' \qquad Q := \bar{u}z.Q'$$

Entonces una interacción entre P y Q se da cuando corremos estos procesos en paralelo:

$$u(x).P' \mid \bar{u}z.Q' \rightarrow P'\{z/x\} \mid Q'$$

La fórmula anterior representa el envío del nombre z a través del canal de nombre u, por parte del proceso Q al proceso P. La formula describe formalmente esta interacción haciendo uso de la relación de reducción $\rightarrow$. Si por un momento llamamos L y R a los procesos denotados por las expresiones a la izquierda y a la derecha del símbolo $\rightarrow$, entonces la fórmula anterior señala que L se reduce a R, donde L denota el proceso compuesto de P y Q antes de la interacción, y R denota lo que resulta después de la interacción. Nótese que luego de recibir el nombre z, el proceso P' renombra su variable x con z, en tanto que luego de enviar z, el proceso Q' queda corriendo en paralelo con P'. En el π-cálculo un proceso L puede reducirse a más de un proceso R. Por ejemplo, si dos procesos concurrentes P_1 y P_2 están esperando

enviar por un mismo canal, y un tercer proceso concurrente Q está escuchando por ese canal, entonces hay dos interacciones posibles, a saber, P_1 con Q, y P_2 con Q. Es decir:

$$\begin{aligned} L &:= u(x).Q' \mid \bar{u}y \mid \bar{u}z \quad \rightarrow \quad Q'\{y/x\} \mid 0 \mid \bar{u}z =: R_1 \\ L &:= u(x).Q' \mid \bar{u}y \mid \bar{u}z \quad \rightarrow \quad Q'\{z/x\} \mid \bar{u}y \mid 0 =: R_2 \end{aligned}$$

En este caso vemos que L se reduce a R_1 y que también L se reduce a R_2. Un caso similar ocurre con la suma de procesos:

$$\begin{aligned} L &:= u(x).Q' \mid (\bar{u}y.P_1 + \bar{u}z.P_2) \quad \rightarrow \quad Q'\{y/x\} \mid P_1 := R_1 \\ L &:= u(x).Q' \mid (\bar{u}y.P_1 + \bar{u}z.P_2) \quad \rightarrow \quad Q'\{z/x\} \mid P_2 := R_2 \end{aligned}$$

Aquí también existen dos interacciones posibles, y por lo tanto hay dos reducciones para L. Nótese, sin embargo, que en el caso de la suma, la interacción se produce con uno de los sumandos mientras que el otro se descarta. Esto no es así en el caso de la composición paralela que acabamos de revisar, donde el proceso que no participaba de la interacción continuaba corriendo en paralelo.

Por último, dentro de la definición de procesos del π-cálculo podemos incorporar funciones en el espacio de nombres. Por ejemplo, supongamos que el espacio de nombres contiene los números naturales y definamos $suc(n) = n + 1$ la función sucesor. Entonces la siguiente expresión:

$$C := (a).(\ \bar{a}\langle \mathbf{0} \rangle \mid !(\ a(x).\bar{u}\langle x \rangle.\bar{a}\langle suc(x) \rangle)\ )$$

Representa un proceso que actúa como contador: cada vez que un proceso distinto a C escucha por el canal de nombre u, éste obtiene el natural igual al número de veces que C ha enviado un número por u. Nótese que se ha ennegrecido el natural $\mathbf{0}$ para diferenciarlo del proceso inerte 0.

Ya hemos visto las nociones básicas que componen el π-cálculo. Corresponde ahora definir formalmente los conceptos que se utilizarán en el modelo de máquinas sociales. Comenzamos definiendo las expresiones de procesos del π-cálculo.

Definición 4.16 *Sea $\mathcal{N}$ un conjunto de nombres. Se define el conjunto $\mathcal{P}$ de expresiones de procesos del π-cálculo mediante la siguiente gramática:*

$$\begin{aligned} P &::= \sum_{i \in I} \pi_i.P_i \ \Big|\ P_1 \mid P_2 \ \Big|\ (a).P \ \Big|\ !P \ \Big|\ A(\bar{x}) \\ \pi &::= w(y) \ \Big|\ \bar{u}z \end{aligned}$$

Donde I es un conjunto de indización finito; $a, w, y, u, z \in \mathcal{N}$; $\bar{x} = (x_1, \ldots, x_n)$ es tal que $x_i \in \mathcal{N}$ para todo $i = 1, \ldots, n$. También, adoptaremos las siguientes convenciones:

1. *El proceso inerte, símbolo 0, se define como la sumatoria vacía;*

2. *Cada vez que el proceso inerte 0 vaya precedido por algún prefijo de input o output, será omitido. De esta forma escribiremos $u(x)$ en vez de $u(x).0$, a la vez que $\bar{u}z$ en vez de $\bar{u}z.0$;*

3. *Si resulta conveniente, anotaremos el prefijo de output como* $P := \bar{u}\langle z\rangle.P'$ *en vez de* $P := \bar{u}z.P'$*;*

4. *Si* $S = \{P_i : i \in I\}$ *es un conjunto de procesos indexados por un conjunto finito* I*, entonces denotaremos la composición de los procesos en* S *por:*

$$\prod_{i\in I} P_i$$

En general llamaremos a las expresiones de procesos del π*-cálculo simplemente* expresiones *del* π*-cálculo,* términos *del* π*-cálculo, o* procesos *del* π*-cálculo.*

Definición 4.17 *Diremos que* $\mathcal{P}$ *es el conjunto de procesos del* π*-cálculo sobre* $\mathcal{N}$*, notación* $\mathcal{P} = \mathcal{P}(\mathcal{N})$*, si de acuerdo a la definición anterior,* $\mathcal{P}$ *es un conjunto de expresiones de procesos del* π*-cálculo con conjunto de nombres* $\mathcal{N}$*.*

Denotaremos por $\mathrm{n}(P)$ el conjunto de todos los nombres que aparecen en P. Diremos que un nombre x está ligado si es parte de una expresión de la forma $u(x).P$ o $(x).P$. Si un nombre no está ligado entonces diremos que es un nombre libre. Denotaremos por $\mathrm{fn}(P)$ los nombres libres de la expresión P.

La única regla sintáctica que hasta el momento no se ha explicado es la definición paramétrica $A(\bar{x})$. En lo que sigue la expresión $(\bar{x})$ denota una tupla de nombres. Una definición paramétrica es un conjunto finito de ecuaciones de la forma:

$$\begin{aligned} D_1(\bar{x}_1) &:= T_1 \\ &\vdots \\ D_n(\bar{x}_n) &:= T_n \end{aligned}$$

tal que cada T_i es una expresión del π-cálculo, y tal que para todo $i = 1, \ldots, n$, todos los nombres libres de T_i aparecen en $\bar{x}_i$. Una instanciación $D_i(\bar{n}_i)$ de $D_i(\bar{x}_i)$ es el término $T_i\{\bar{n}_i/\bar{x}_i\}$. Informalmente hablando, la definición paramétrica permite tomar una expresión del π-cálculo cualquiera y utilizar los nombres libres que en ella ocurren como parámetros. Veamos esto con un ejemplo. Considérese el siguiente proceso:

$$T := a(x).\bar{b}x$$

y supongamos que queremos definir el proceso:

$$T' := c(x).\bar{d}x$$

y que más adelante también necesitaremos el proceso:

$$T'' := u(x).\bar{v}x$$

Entonces resultaría muy útil poder parametrizar esta expresión. Eso es precisamente lo que hace la definición paramétrica. De esta forma la expresión

$$A(a, b) := a(x).\bar{b}x$$

Nos permite tener una expresión que como argumento recibe el nombre del canal por el que se escucha, y el nombre del canal por el que se recibe. De esta forma:

$$A(a,b) := a(x).\bar{b}x$$
$$A(c,d) := c(x).\bar{d}x$$
$$A(u,v) := u(x).\bar{v}x$$

Y en general, $A(n_1, n_2) = T\{n_1/a, n_2/b\}$. Pero eso no es todo. También la definición paramétrica permite definir un proceso en términos de otro, como en el siguiente ejemplo:

$$C(u,v) := u(x).D(x,v) + v(y).D(y,u)$$
$$D(a,b) := \bar{b}a$$

Y también permite definir procesos de forma recursiva, como por ejemplo:

$$P(a,b) := a(x).\bar{b}x.P(b,a)$$

Ahora que ya hemos visto como se definen las expresiones del π-cálculo, y revisado cada una de las reglas sintácticas y su significado, corresponde definir formalmente las nociones de interacción y comunicación entre procesos. Como ya se mencionó en algunos párrafos más arriba, estas nociones se recogen formalmente en el π-cálculo por medio de la relación de reducción. Sin embargo, para definir formalmente esta relación necesitamos antes introducir el concepto de congruencia estructural, que permite hacer equivalentes expresiones que aún pudiendo ser sintácticamente distintas representan un mismo proceso. A continuación se explica informalmente lo que es una relación de congruencia; la definición formal se omite.

Una relación de congruencia $\equiv$ en un conjunto de expresiones A, es una relación de equivalencia en A que satisface que, cada vez que se tiene $P \equiv P'$ entonces toda expresión Q que *contenga* a P es equivalente a la expresión Q' resultante de reemplazar P' por P en Q. Por ejemplo si se tiene $P \equiv P'$, entonces las expresiones:

$$\begin{aligned} Q &:= (a).(\ u(x).P + v(y).M\ ) \\ Q' &:= (a).(\ u(x).P' + v(y).M\ ) \end{aligned}$$

Satisfacen $Q \equiv Q'$. En general, esto puede pensarse como que si P y P' representan lo mismo, aún cuando pudieran tener distinta sintaxis, entonces cualquier expresión que se construya utilizando P representa lo mismo que la que se construye exactamente igual pero usando P' en vez de P. A continuación se define la relación de congruencia estructural.

Definición 4.18 *Se define la relación de* congruencia estructural*, símbolo $\equiv$, como la congruencia más pequeña sobre $\mathcal{P}$ que satisface las siguientes ecuaciones:*

1. *$P \equiv Q$ cada vez que P sea α-convertible a Q;*
2. *$P|0 \equiv P, \quad P|Q \equiv Q|P, \quad P|(Q|R) \equiv (P|Q)|R$;*
3. *$!P \equiv P|!P$;*
4. *$(a).\,0 \equiv 0, \quad (a).(b).P \equiv (b).(a).P$;*

5. $(a).(P|Q) \equiv P|(a).Q$ *si* $a \notin \text{fn}(P)$;

6. $(a).a(x).P \equiv 0, \quad (a).\bar{a}v.P \equiv 0$;

7. $(a).!a(x).P \equiv 0, \quad (a).!\bar{a}v.P \equiv 0.$

De esta forma, cualquier par de expresiones Q y Q' serán congruentes estructuralmente si, aplicando cero o más veces cualquiera de las reglas establecidas en la definición, en cualquier orden y en cualquier subexpresión, se puede transformar Q en Q'. A continuación un ejemplo:

$$\begin{aligned}
P &:= (a).(\ 0|(0|(a).\bar{a}v.Q)\ ) \\
P &\equiv (a).(\ 0|((a).\bar{a}v.Q|0)\ ) \\
P &\equiv (a).(\ 0|(a).\bar{a}v.Q\ ) \\
P &\equiv (a).(\ (a).\bar{a}v.Q|0\ ) \\
P &\equiv (a).(\ (a).\bar{a}v.Q\ ) \\
P &\equiv (a).0 \\
P &\equiv 0
\end{aligned}$$

Ahora que ya se ha presentado el concepto de congruencia estructural, se puede definir formalmente la relación de reducción cuya idea intuitiva revisamos con anterioridad.

Definición 4.19 *Se define la relación de* reducción*, símbolo* $\rightarrow$*, como la relación más pequeña que satisface las siguientes reglas:*

COM: $$(\ x(y).P\ +\ M\ )\,|\,(\ \bar{x}z.Q\ +\ N\ )\ \rightarrow\ P\{z/y\}\,|\,Q$$

PAR: $$\frac{P \rightarrow P'}{P\,|\,Q \rightarrow P'\,|\,Q}$$

RES: $$\frac{P \rightarrow P'}{(y)P\ \rightarrow\ (y)P'}$$

STRUCT: $$\frac{Q \equiv P \quad P \rightarrow P' \quad P' \equiv Q'}{Q\ \rightarrow\ Q'}$$

DEF: $$\frac{P\{\bar{b}/\bar{a}\} \rightarrow P'}{A(\bar{b})\ \rightarrow\ P'} \quad \text{si } A(\bar{a}) := P$$

Denotaremos por $\rightarrow^*$ *la clausura refleja y transitiva de* $\rightarrow$*.*

Por último presentamos una definición muy útil que nos permitirá caracterizar el conjunto de nombres de canales utilizados por un proceso del π-cálculo.

Definición 4.20 *Sea $\mathcal{N}$ un conjunto de nombres, $\mathcal{P} = \mathcal{P}(\mathcal{N})$ el conjunto de procesos del π-cálculo sobre $\mathcal{N}$ y $\mathcal{N}_0 \subseteq \mathcal{N}$. Sea $A \in \mathcal{P}$ un proceso cualquiera. Diremos que A* tiene sus canales de interacción en *$\mathcal{N}_0$, notación $\mathcal{K}(A) \subset \mathcal{N}_0$, si y solo si se satisfacen lo siguiente:*

1. *Si $A := \sum_{i \in I} \pi_i.P_i$, entonces para todo $i \in I$:*
   (a) *Si $\pi_i := x(y)$, entonces $x \in \mathcal{N}_0 \wedge$ para todo $z \in \mathcal{N}$, $\mathcal{K}(P_i\{z/y\}) \subset \mathcal{N}_0$;*
   (b) *Si $\pi_i := \bar{x}y$, entonces $x \in \mathcal{N}_0 \wedge \mathcal{K}(P_i) \subset \mathcal{N}_0$;*
2. *Si $A := P_1 \mid P_2$, entonces $\mathcal{K}(P_1) \subset \mathcal{N}_0 \wedge \mathcal{K}(P_2) \subset \mathcal{N}_0$;*
3. *Si $A := (a).P$, entonces $\mathcal{K}(P) \subset \mathcal{N}_0$;*
4. *Si $A := !P$, entonces $\mathcal{K}(P) \subset \mathcal{N}_0$;*
5. *Si $A := A(\bar{x})$ tal que $\bar{x}$ es de largo n, entonces para todo $\bar{z} \in \mathcal{N}^n$, $\mathcal{K}(A(\bar{z})) \subset \mathcal{N}_0$.*

Analicemos brevemente esta definición. La idea principal es que un proceso A tiene sus canales de interacción en $\mathcal{N}_0$ si y solo si los nombres de los canales que usa para comunicarse, así como cualquiera de los subprocesos que lo componen o en los que pueda derivar, están en el conjunto $\mathcal{N}_0$. La primera condición abarca el caso en que A es una sumatoria. Si un sumando espera recibir un nombre por un canal, ese canal debe tener su nombre en $\mathcal{N}_0$, y además cualquier proceso al que pase luego de escuchar por ese canal, debe tener sus canales de interacción en $\mathcal{N}_0$. Si el sumando espera enviar un nombre por un canal, entonces ese canal debe tener su nombre en $\mathcal{N}_0$, aunque no es necesario que el nombre que se envía por el canal, es decir el mensaje, esté en $\mathcal{N}_0$. La segunda condición señala que si un proceso está compuesto de dos procesos corriendo en paralelo entonces cada proceso de tener sus canales de interacción en $\mathcal{N}_0$. La tercera condición y la cuarta responden al caso de la restricción y la replicación. Finalmente la última condición establece que si un proceso se define de forma paramétrica, entonces no importa qué argumentos reemplacen los parámetros, siempre el proceso resultante debe tener sus nombres en $\mathcal{N}_0$.

Con esto finalizamos la exposición de las nociones preliminares del π-cálculo que resultan elementales para la comprensión del modelo de máquinas sociales. El lector que se desee profundizar más en el π-cálculo encontrará un análisis más detallado en Milner (1990) y Pierce (1995). Para una exposición completa de la teoría del π-cálculo se recomienda Milner (1999).

Capítulo 5

Modelo formal de máquinas sociales

A continuación se presenta el modelo formal de máquinas sociales elaborado durante la memoria. Una máquina social será modelada como un dispositivo computacional que tiene un *canal de entrada* por donde espera recibir un *input*, y uno o más *canales de salida* los que usa para el envío de *outputs*. Una vez que la máquina ha recibido un input desde el *entorno*, ésta lleva a cabo una secuencia de *micropasos de computación* uno tras otro, los que terminan generando los distintos outputs que se enviarán por cada canal de salida. La actividad computacional que mantiene la máquina social desde que ingresa un input recibido desde el entorno, hasta que se computa cada output a ser enviado por los canales de salida se denominará *macropaso de computación*. De esta forma, la máquina recibe un input por su canal de entrada, realiza un macropaso de computación, envía cada output por el canal de salida correspondiente, y se dispone a recibir un nuevo input desde el entorno. Este entorno con que la máquina social interactúa está conformado por personas y posiblemente otras máquinas sociales. Como veremos más adelante, el modelo de máquinas sociales que se presenta permite discriminar para cada interacción la identidad de los emisores y receptores que participan de ella; en particular, cuando la máquina interactúa con su entorno, puede determinar exactamente con quién lo está haciendo. De esta forma, la máquina social recibe por su canal de entrada un input de un cierto emisor, y posteriormente envía por sus canales de salida un output a cada uno de los receptores que están *conectados* a la máquina escuchando por ese canal. La máquina recibe por su canal de entrada un solo input, el primero que haya sido enviado desde el entorno, realiza un macropaso, y luego dispone simultáneamente en sus canales de salida los outputs respectivos. Un aspecto fundamental de las máquinas sociales es que, en cada momento, la máquina social almacena un estado bajo la forma de un *string de estado* determinado. Como veremos, este string de estado persiste entre dos macropasos de computación, de modo que cada vez que la máquina comienza un macropaso, su string de estado es exactamente igual al que tenía al término del macropaso anterior. En específico, cada vez que este dispositivo recibe un cierto input, y dado un string de estado inicial w_0, se proceden a computar los outputs a enviar por los canales de salida, así como también el nuevo string de estado w_1 de la máquina. La computación que se lleva a cabo es determinista, de modo que los outputs y el nuevo string de estado w_1 dependen exclusivamente del input y el string de estado inicial w_0. Ahora bien, cuando más adelante la máquina reciba un nuevo input, entonces el string de estado inicial de la máquina será

justamente w_1, es decir, el string de estado final que se había calculado en la computación anterior. En base a ese string de estado y el nuevo input se generarán los outputs y otro nuevo string de estado w_2 correspondiente. El concepto fundamental de computación que subyace en este tipo de comportamiento es el de *persistencia*. Esta persistencia es introducida en el modelo de máquina social mediante la posibilidad de guardar un string de estado desde el término de una computación al inicio de la siguiente. Para incorporar formalmente la noción de persistencia en el modelo de computación de máquinas sociales se ha seguido el enfoque de máquinas persistentes de Turing desarrollado por Goldin, Smolka, Attie y Sonderegger (2004).

La actividad computacional que comprende un macropaso, es decir, aquella que se inicia cuando la máquina ingresa el input, que continúa con una secuencia de micropasos de computación, y que —posiblemente— termina generando los outputs respectivos a cada canal de salida, la denominaremos *comportamiento interno* de la máquina. Para modelar este comportamiento interno usaremos un formalismo muy similar al de la máquina de Turing de n cintas, con la diferencia que tendremos especial cuidado de incorporar en tal formalismo desde un comienzo esta idea de canales de comunicación que más adelante nos permitirán formalizar las nociones de interacción entre máquinas y personas. De esta manera, una máquina social estará formada por múltiples cintas; una cinta se utilizará para colocar el input que proviene del entorno, otra cinta almacenará el string de estado, y las restantes almacenarán el output a ser enviado al entorno por cada canal. Para desarrollar esta parte del modelo seguiremos el enfoque que Lewis y Papadimitriou (1981) adoptan al introducir el formalismo de máquinas de Turing. Una vez que contemos con el modelo interno de máquinas sociales y con la noción de computación persistente, nos ocuparemos de modelar el *comportamiento externo* que las máquinas sociales exhiben en entornos compuestos por personas y otras máquinas. Este modelo externo se llevará a cabo utilizando el formalismo de π-cálculo que se introdujo en la Sección 4.4, y nos permitirá modelar adecuadamente la interacción entre las máquinas sociales y las personas.

La presentación del modelo se organiza en torno a estas tres grandes áreas que hemos indicado hasta ahora. Primero, presentamos el modelo interno de máquinas sociales. Luego, introducimos la noción de computación persistente. Finalmente, modelamos el comportamiento externo de las máquinas sociales y deducimos expresiones formales dentro del modelo que permitan dar cuenta de algunas de las principales propiedades de las máquinas sociales. Cada una de estas tres áreas se dispone como una sección aparte del modelo.

5.1. Comportamiento interno de máquinas sociales

Vamos a modelar una máquina social como un dispositivo conformado por un número finito de *cintas* y una *unidad de control*. Cada cinta consiste en una sucesión infinita de *celdas* que tiene comienzo pero no fin, extendiéndose hacia la derecha tanto como se quiera. La máquina opera sobre cierto alfabeto Σ, y en cada celda de una cinta se encuentra un símbolo de Σ, siendo el *símbolo blanco* # el que se encuentra por defecto en toda celda que no haya sido examinada anteriormente. Para cada una de sus cintas, la máquina social dispone de un *cabezal* que se posiciona en una de las celdas de esa cinta y le permite leer su

contenido. En cada etapa de la computación, la unidad de control almacena un *estado interno* de la máquina; uno dentro de un conjunto fijo y finito de estados posibles que se definen al mismo tiempo que se define cada máquina —estos estados internos intervienen en la sucesión de micropasos de una computación, y no guardan relación con el string de estado que persiste durante macropasos, del que nos ocupamos en la siguiente sección—. En cada *micropaso de computación* la unidad de control lee los símbolos bajo los cabezales, y según lo que lea y el estado en que esté, pasa a un nuevo estado interno y lleva a cabo una acción en cada una de sus cintas. Estas acciones pueden ser: escribir un nuevo símbolo en la celda que está siendo examinada por el cabezal; desplazar el cabezal hacia la izquierda; o bien desplazar el cabezal hacia la derecha. Si la máquina intenta mover algún cabezal a la izquierda de la primera celda de una cinta, entonces la computación se interrumpe y decimos que la máquina se *cuelga*. La máquina intenta pasar continuamente de una etapa de computación a otra hasta llegar a un *estado de halting* o *estado de parada*, que significa el fin del cómputo. Si esto ocurre, decimos que la máquina *converge*. Si no, entonces decimos que *diverge*.

Cada cinta de una máquina social cumple una función bien definida en el modelo de máquinas sociales. En total, éstas se componen de una *cinta de entrada*, una *cinta de trabajo*, y una o más *cintas de salida*. La cinta de entrada almacena el input que la máquina recibe por su canal de input. La cinta de trabajo es una cinta interna que la máquina usa para llevar a cabo sus cómputos; además, esta cinta almacena la palabra de estado de la máquina —lo que será detallado en la siguiente sección—. Finalmente, por cada canal de salida de la máquina existe una cinta de salida, que almacena el output que espera ser enviado por el canal respectivo. Junto con lo anterior, una observación muy importante es que los nombres de los canales de salida de una máquina social son parte también de su alfabeto. Esto es, si $\lambda_1, \ldots, \lambda_n$ son los nombres de los canales de salida, entonces $\lambda_1, \ldots, \lambda_n \in \Sigma$. También, cada una de las cintas de la máquina social está asociada a un identificador. En el caso de la cinta de entrada, su identificador es el símbolo $\triangleright$; en el caso de la cinta de trabajo, su identificador es el símbolo $\square$; y en el caso de cada cinta de salida, su identificador es el propio nombre del canal de salida asociado. Estos identificadores nos permitirán nombrar las cintas fácilmente; así, por ejemplo, la cinta de salida asociada al canal de salida de nombre λ será referida simplemente como la cinta λ. La Figura 5.1 ilustra este modelo de máquinas sociales. Como vemos en esta figura, cada cinta va precedida por su símbolo identificador. Así, desde arriba hasta abajo tenemos que la primera cinta es la cinta de entrada, la segunda es la cinta de trabajo, y las n restantes son las cintas de salida. La unidad de control se ecuentra en el estado q, y los cabezales de las cintas se encuentran sobre las celdas que han sido resaltadas con doble borde, es decir, están sobre las celdas de contenido $b, k, \lambda_n, \ldots, \#$. Nótese que la cinta de salida λ_1 tiene en su primera celda el símbolo λ_n, que también corresponde a un nombre de canal de salida. Más adelante reconoceremos la importancia de que la máquina opere sobre un alfabeto que incluya los nombres de los canales de salida. Como un primer acercamiento, si en la Figura 5.1 el estado interno fuera de parada, podríamos interpretar el output de λ_1 como un mensaje de «ola» por parte del agente conectado al canal λ_n al agente conectado al canal λ_1.

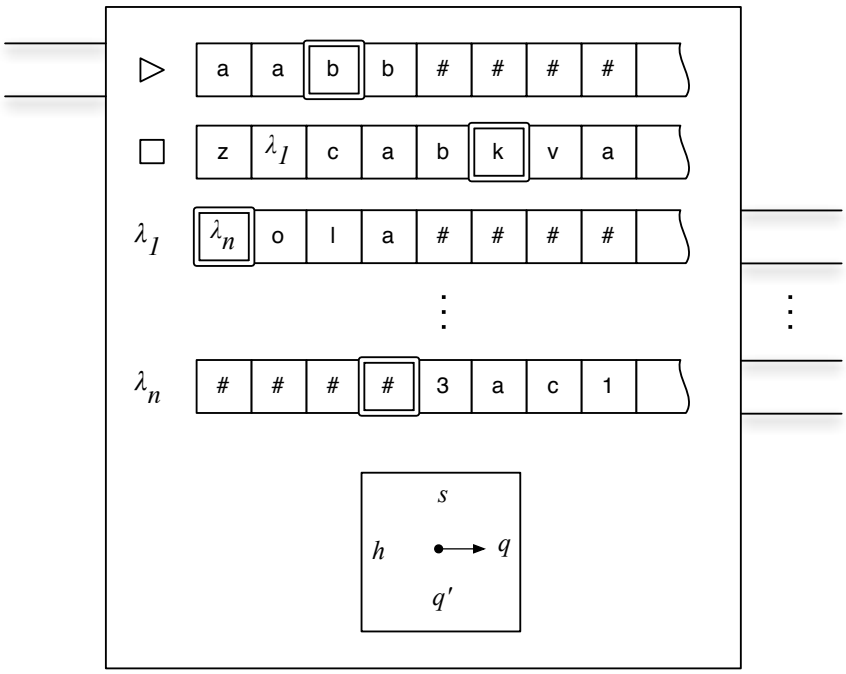


Figura 5.1: Ilustración de una máquina social.

Definición 5.1 *Una* Máquina Social *es una tupla* $M = (Q, \Gamma, \Lambda, \delta, s, H)$*, donde*

- Q *es un conjunto finito de* estados*;*
- Γ *es un alfabeto finito tal que* $\# \in \Gamma$ *y* $\{L, R, \triangleright, \square\} \cap \Gamma = \emptyset$*;*
- Λ *es el conjunto de* nombres de canales de salida *tal que* $\Lambda \cap \{\#, L, R, \triangleright, \square\} = \emptyset$ *y es no vacío. Además se define* $\Lambda_0 = \Lambda \cup \{\triangleright, \square\}$ *y* $\Sigma = \Lambda \cup \Gamma$*;*
- $s \in Q$ *es el* estado inicial*;*
- $H \subseteq Q$ *es el conjunto de* estados de parada*;*
- $\delta : (Q - H) \times (\Lambda_0 \to \Sigma) \nrightarrow Q \times (\Lambda_0 \to (\Sigma \cup \{L, R\}))$ *es la* función parcial de transición.

Revisemos brevemente esta definición. Q es el conjunto finito de estados internos en que puede encontrarse la unidad de control. Cuenta con un estado especial s, el estado inicial, que corresponde al estado con que comienza cada computación. El conjunto de estados H corresponde a los estados de halting; una vez que la máquina social pasa a una etapa de computación cuyo estado está en H, la máquina termina su computo. Luego tenemos los alfabetos Γ y Λ. El alfabeto Λ es el conjunto de nombres de canales de salida, mientras que el alfabeto Γ representa el *conjunto básico de símbolos de comunicación* también llamado *conjunto base de comunicación*. Este conjunto básico se extiende con los nombres de canales

de salida al *alfabeto extendido* o *alfabeto de máquina* $\Sigma = \Lambda \cup \Gamma$, que es el conjunto de los símbolos con que opera la máquina. Más adelante, cuando examinemos la interacción entre máquinas y agentes, veremos cuánto facilita separar el alfabeto de máquina en un alfabeto común a todos los actores y otro que contiene símbolos exclusivos de cada máquina. También, Γ debe contener el símbolo blanco $\#$, ya que como se ha comentado es el símbolo que se encuentra por defecto en las celdas. En cambio, el alfabeto Λ no debe contenerlo, ya que si así fuese no podríamos discriminar entre el símbolo blanco y un canal. Por su parte, ambos alfabetos no deben incluir los símbolos que identifican las cintas de entrada y trabajo, ni los símbolos L y R, cuyo uso está reservado para la función parcial de transición. Por último, usamos el símbolo Λ_0 para denotar el conjunto de todos los símbolos identificadores de cintas.

La función parcial de transición δ indica a la unidad de control cómo pasar de una etapa de computación a otra. Cada par ordenado $(q, c) \in Dom(\delta)$ está compuesto por un estado interno q de la máquina, y una asignación c que a cada identificador de cinta le asigna el símbolo que lee el cabezal respectivo. Es decir, c representa el contenido que están leyendo los cabezales. Una asignación $\delta(q, c) = (p, act)$ representa una instrucción de la unidad de control que le indica pasar del estado interno q con sus cabezales leyendo según c, al estado interno p realizando por cada cinta de identificador $\lambda \in \Lambda_0$ una de las tres acciones que mencionamos con anterioridad: reemplazar por $act(\lambda)$ el contenido de la celda bajo el cabezal, si $act(\lambda) \in \Sigma$; mover el cabezal a la izquierda si $act(\lambda) = L$; o moverse el cabezal a la derecha si $act(\lambda) = R$. Ahora bien, en el caso en que una máquina social se encuentre en una etapa de computación que no esté considerada por la función parcial de transición, entonces diremos que la máquina se *cuelga* —de la misma forma que cuando intentábamos mover un cabezal a la izquierda de la primera celda de una cinta—. En la mayoría de los casos intentaremos no definir la función de transición δ utilizando notación funcional, sino que usaremos representaciones más intuitivas como tablas y diagramas de estados. Establecemos esto en la siguiente observación.

Observación 5.2 Representaremos la función parcial de transición de una máquina social $M = (Q, \Gamma, \Lambda, \delta, s, H)$ usando tablas o diagramas de estado. Cuando definamos la función de transición por una tabla, fijaremos un orden en Λ_0 para así escribir las funciones c y act que participan de una asignación $\delta(q, c) = (p, act)$, en forma de secuencias de símbolos. Habrá una fila cabecera que será la encargada de indicar cuál es ese orden, y por lo tanto permitirá interpretar correctamente el resto de la tabla. Además, las filas de la tabla estarán divididas en dos secciones, la sección izquierda de cada fila representa un elemento del dominio de δ, mientras que la sección derecha representa la imagen del elemento correspondiente. Por ejemplo, considérese la función de transición definida por la siguiente tabla:

q	$\triangleright$	$\square$	λ_1	q'	$\triangleright$	$\square$	λ_1
s	$\#$	1	0	h	$\#$	R	1

Tabla 5.1: Notación tabular.

Aquí, la función de transición cuenta con un único elemento, descrito por la ecuación $\delta(s, c) = (h, act)$ donde $c = \{(\triangleright, \#), (\square, 1), (\lambda_1, 0)\}$ y donde $act = \{(\triangleright, \#), (\square, R), (\lambda_1, 1)\}$. Como se observa, la fila cabecera permite reconstruir las funciones c y act asignando a cada identificador de cinta el símbolo que se escribe bajo la columna respectiva. También usaremos diagramas de estados para definir la función de transición δ. Este caso es similar a la notación tabular, en cuanto fijamos un orden para los

identificadores de cinta, y por cada entrada $\delta(q, c) = (p, act)$ de la función de transición dibujamos una flecha desde un nodo q a un nodo p, y la etiquetamos con $(c(\lambda_0) \dots c(\lambda_n)\ ,\ act(\lambda_0) \dots act(\lambda_n))$. De esta forma, si ordenamos los identificadores anteriores como $\triangleright, \square, \lambda_1$, entonces la función de transición descrita en la Tabla 5.1 puede también definirse mediante el diagrama de estados descrito en la Figura 5.2.

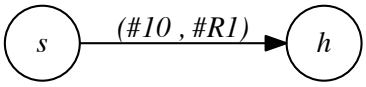


Figura 5.2: Notación como diagrama de estados.

Ejemplo 5.3 A continuación presentamos un ejemplo sencillo de máquina social. Sea una máquina social M definida según $M = (Q, \Gamma, \Lambda, \delta, s, H)$ de modo que:

$$Q = \{s, q, h\}$$
$$\Gamma = \{\#, 1, 0\}$$
$$\Lambda = \{\lambda_1, \lambda_2\}$$
$$H = \{h\}$$

En donde la función de transición se define de acuerdo a la siguiente tabla:

q	$\triangleright$	$\square$	λ_1	λ_2	q'	$\triangleright$	$\square$	λ_1	λ_2
s	$\#$	$\#$	$\#$	$\#$	q	$\#$	R	1	0
q	$\#$	$\#$	1	0	h	$\#$	L	R	R

Dada esta máquina social, tenemos que el alfabeto de máquina y el de identificadores de cintas son:

$$\Sigma = \{\#, 1, 0, \lambda_1, \lambda_2\}$$
$$\Lambda_0 = \{\triangleright, \square, \lambda_1, \lambda_2\}$$

Como podemos observar de la función de transición, esta máquina está diseñada para leer en todos sus cabezales el símbolo blanco al comienzo de la computación, escribir en sus cintas de salida λ_1 y λ_2 los símbolos 1 y 0 respectivamente, y terminar con los cabezales de esas cintas una posición a la derecha de como empezaron. La cinta de input no se modifica, lo que se logra escribiendo el mismo símbolo que se lee en cada etapa. Por último la cinta de trabajo desplaza su cabezal a la derecha y luego a la izquierda, sin afectar su contenido. Nótese que la máquina no puede comenzar a computar si no lee símbolos blancos en sus cabezales.

Ahora que ya se ha establecido la definición formal de máquina social corresponde formalizar las nociones de etapa de computación y micropaso de computación. Al igual que en el caso de máquinas de Turing, vamos a recoger estas nociones formalmente bajo el concepto

de *configuración* y de la relación binaria *lleva en un micropaso*, definida sobre el conjunto de tales configuraciones. Una configuración de la máquina social estará determinada completamente por el estado interno de la unidad de control de la máquina, el contenido de cada una de sus cintas y la posición de los cabezales en cada cinta.

Definición 5.4 *Una* configuración *de una máquina social* $M = (Q, \Gamma, \Lambda, \delta, s, H)$ *es un elemento del conjunto:*

$$\mathcal{C}_M \quad = \quad Q \quad \times \quad \Lambda_0 \to (\Sigma^* \times \Sigma \times (\Sigma^*(\Sigma - \{\#\}) \cup \varepsilon)).$$

Dada una configuración $(q, l) \in \mathcal{C}_M$ *y un símbolo* $\lambda \in \Lambda_0$*, usaremos la notación* $l(\lambda) = u\underline{a}v$ *como alternativa de* $l(\lambda) = (u, a, v)$ *cuando resulte más conveniente. También nos permitiremos fijar un orden de* Λ_0 *y escribir una configuración de la forma* (q, l) *como* $(q, u_1\underline{a_1}v_1, \ldots, u_n\underline{a_n}v_n)$*. Finalmente, llamaremos* configuración de inicio *a una configuración* $(q, l) \in \mathcal{C}_M$ *tal que* $q = s$*, y llamaremos* configuración de parada *a una configuración* $(q, l) \in \mathcal{C}_M$ *que satisfaga* $q \in H$*.*

Una configuración de una máquina social consta de dos componentes. La primera componente corresponde al estado interno en que se encuentra la unidad de control de la máquina. La segunda, corresponde a una función que a cada indentificador de cinta le asocia un contenido. El contenido de cada cinta es un elemento de la forma $\Sigma^* \times \Sigma \times (\Sigma^*(\Sigma - \{\#\}) \cup \varepsilon)$. Esto significa que el contenido tiene a su vez tres componentes: el string formado por los símbolos anteriores a la celda donde se ubica el cabezal; el símbolo de la celda sobre el que está el cabezal; y el string formado por los símbolos de las celdas que siguen a la que está siendo examinada por el cabezal. Dado que las cintas crecen sin límites a la derecha y que el blanco # se halla en toda celda que no ha sido examinada, la tercera componente del contenido de una cinta se especifica por una palabra vacía o una que termina con símbolo distinto de blanco; de esta forma, entenderemos que a la derecha de esta tercera componente, todas las celdas contienen el blanco #. Además, como se indica en la definición, si el contenido de una cinta es $(abra, c, adabra)$ entonces también lo anotaremos como $abra\underline{c}adabra$, y ocuparemos también esta notación para expresar la propia configuración de la máquina, fijando para ello un orden en los identificadores de Λ_0. Veamos esto con un ejemplo.

Ejemplo 5.5 En este ejemplo vamos a determinar la configuración de la máquina social que se ilustra en la Figura 5.1. Para ello, vamos a asumir que en cada cinta, todos los símbolos de las celdas a la derecha de la última celda son blancos #. Además supondremos que la máquina tiene solo dos cintas de salida, λ_1 y λ_n. De esta forma, si ordenamos Λ_0 como $\triangleright, \square, \lambda_1, \lambda_n$, entonces la configuración de la máquina social de la Figura 5.1 se puede escribir como:

$$(q\,,\ aa\underline{b}b\,,\ z\lambda_1 cab\underline{k}va\,,\ \underline{\lambda_n}ola\,,\ ####\underline{\#}3ac1)$$

Ya contamos con la noción de configuración, por lo que procedemos a definir la relación binaria *lleva en un micropaso* sobre el conjunto de las configuraciones. Al igual que en el caso de máquinas de Turing, esta relación especifica cuáles son los pasos de computación

posibles de una máquina. Esto es, la relación señala cuándo la máquina podrá pasar de una configuración a otra o, informalmente hablando, de una etapa de computación a otra. En nuestro caso, debido a que las máquinas sociales interactúan con su entorno, debemos ser claros en hacer notar que la noción de *micropaso* forma parte del comportamiento interno de la máquina, es decir, los micropasos no son visibles desde fuera de la máquina social.

Definición 5.6 *Sea $M = (Q, \Gamma, \Lambda, \delta, s, H)$ una máquina social. Se define la relación* lleva en un micropaso, *$\vdash_M \subseteq \mathcal{C}_M \times \mathcal{C}_M$, de la siguiente forma:*

$$(q_1, l_1) \vdash_M (q_2, l_2)$$

si y solo si, siendo $c : \Lambda_0 \to \Sigma, c = \pi_2 \circ l_1$ la función que describe el contenido de los cabezales en la configuración inicial, existe $act : \Lambda_0 \to \Sigma \cup \{L, R\}$ tal que $\delta(q_1, c) = (q_2, act)$ y para todo $\lambda \in \Lambda_0$ se cumple una de las siguientes propiedades. Sea $l_1(\lambda) = (u, a, v)$ y $l_2(\lambda) = (u', a', v')$. Entonces:

1. *$act(\lambda) \in \Sigma$, $u = u'$, $v = v'$, y $a' = act(\lambda)$;*
2. *$act(\lambda) = L$, $u = u'a'$ y se cumple ya sea:*
   (a) *$v' = av$, si $a \neq \# \vee v \neq \varepsilon$, o bien*
   (b) *$v' = \varepsilon$, si $a = \# \wedge v = \varepsilon$;*
3. *$act(\lambda) = R$, $u' = ua$, y se cumple ya sea:*
   (a) *$v = a'v'$, o bien*
   (b) *$v = v' = \varepsilon \wedge a' = \#$.*

Revisemos brevemente esta definición. Primero que todo, para que una configuración lleve en un micropaso a otra, el estado interno de la unidad de control y la función que describe el contenido de los cabezales deben pertenecer al dominio de la función parcial de transición δ. Si esto no ocurre, diremos que la máquina se *cuelga*. Ahora, si esto sí sucede, la configuración llevará en un micropaso a otra satisfaciendo en cada cinta una de las siguientes condiciones:

1. La unidad de control escribe el símbolo σ bajo el cabezal, si el valor de la función de transición para la cinta es $\sigma \in \Sigma$;

2. La unidad de control desplaza a la izquierda el cabezal de la cinta, si el valor de la función de transición para la cinta es L. Además, si el contenido inicial de la cinta (u, a, v) es tal que $a = \#$, $v = \varepsilon$, entonces al moverse a la izquierda el cabezal, el blanco $\#$ pasa a formar parte de la sucesión de blancos que se encuentran por defecto en las celdas no descritas por el contenido de la cinta. Por eso, el contenido final de la cinta en ese caso es (u', a', v') con $v' = \varepsilon$.

3. La unidad de control desplaza a la derecha el cabezal de la cinta, si el valor de la función de transición para la cinta es R. Además, si el contenido de la cinta (u, a, v) es tal que $v = \varepsilon$, entonces al moverse a la derecha el cabezal estará sobre una celda que

por defecto contendrá el símbolo blanco #, con lo que el contenido final de la cinta en ese caso será (u', a', v') tal que $a' = \#$, $v' = \varepsilon$.

Nótese que en la Definición 5.6, punto número *2*, si (u, a, v) es tal que $u = \varepsilon$, es decir, el cabezal lee la primera celda de una cinta, entonces la ecuación $u = u'a'$ se transforma en $\varepsilon = u'a'$. Ahora bien, no existe ningún par $u' \in \Sigma^*, a' \in \Sigma$ que satisfaga esa ecuación debido a la sencilla razón que $u'a'$ es de largo al menos 1, mientras que ε tiene largo 0. Esto significa, por tanto, que cualquier configuración que deba mover un cabezal a la izquierda de la primera celda de una cinta, simplemente no llevará en un micropaso a ninguna otra configuración. Así, las dos formas que tiene una máquina de colgarse son: o bien que la función de transición no esté definida para el estado interno actual y los símbolos presentes bajo los cabezales; o bien que la unidad de control intente mover un cabezal a la izquierda de la primera celda de una cinta.

A continuación definimos la noción de *computación* de una máquina social, y terminamos la sección ofreciendo un ejemplo de este concepto.

Definición 5.7 *Dada una máquina social $M = (Q, \Gamma, \Lambda, \delta, s, H)$, el símbolo $\vdash_M^*$ denota la clausura refleja y transitiva de $\vdash_M$; diremos que una configuración C_1* lleva en cero o más micropasos *a una configuración C_2 si $C_1 \vdash_M^* C_2$. Dado $n \geq 0$, una* computación *de M de n micropasos es una secuencia de configuraciones $(C_0, C_1, \ldots, C_n) \in \mathcal{C}_M^{n+1}$ tal que:*

$$C_0 \vdash_M C_1 \vdash_M \ldots \vdash_M C_n$$

Diremos que M se detiene *al partir con las cintas en l si (s, l) lleva en cero o más micropasos a una configuración de parada.*

Como vemos, el concepto de computación de una máquina social es análogo al de computación de una máquina de Turing —ver Definición 4.9—. La diferencia que existe entre ambas nociones es que, mientras que la máquina de Turing cuenta con solo una noción de paso de computación, la máquina social cuenta con dos nociones de paso de computación, a saber, el *micropaso* y el *macropaso*. En el caso de la máquina de Turing, el paso de computación representa la acción atómica que puede efectuarse sobre una configuración; en nuestro caso, esa acción atómica la llamamos *micropaso*; el concepto de *macropaso* lo consideramos en la sección siguiente. Finalizamos esta sección con un ejemplo de computación; antes, hacemos una observación sobre ciertas convenciones que adoptaremos para escribir de forma más sucinta la función parcial de transición de las máquinas sociales.

Observación 5.8 En general, al usar notación tabular para representar la función de transición resulta tedioso especificar cada una de las filas correspondientes, especialmente cuando varios elementos del dominio comparten una y la misma imagen. Es por ello que adoptaremos ciertas convenciones para facilitar el uso de la notación tabular. Primero que todo, introducimos el símbolo $*$ en la notación, el que podrá ser usado debajo de un identificador de cinta en la sección izquierda de cada fila. Este símbolo se interpreta como sigue. Si una fila contiene el símbolo $*$ bajo un identificador λ, entonces existe una fila en la tabla por cada símbolo $a \in \Gamma$, con $*$ reemplazado por a. Intuitivamente esto quiere decir que no importa qué símbolo de Γ está leyendo ese cabezal para ese paso de computación. De esta forma, la siguiente tabla:

q	$\triangleright$	$\square$	λ_1	q'	$\triangleright$	$\square$	λ_1
q_1	$\#$	$\#$	$*$	q_2	a	b	c

Tabla 5.2: Convenciones de la notación tabular: símbolo $*$.

Indica que si la máquina tiene estado q_1 y lee el símbolo blanco en sus cintas de entrada y trabajo, entonces actuará según se indica en la sección derecha, para cualquier símbolo en Γ que pueda leer en su cinta de salida. También introducimos el símbolo ρ, que funciona como el símbolo $*$ pero puede ser usado en cualquier sección de la fila. Esto es, dada una fila que tenga el símbolo ρ, por cada símbolo a en Γ, se incluye una fila en la tabla con ρ reemplazado por a. Por ejemplo, la siguiente tabla:

q	$\triangleright$	$\square$	λ_1	q'	$\triangleright$	$\square$	λ_1
q_1	$\#$	$\#$	ρ	q_2	ρ	b	c

Tabla 5.3: Convenciones de la notación tabular: símbolo ρ.

Indica que si la máquina tiene estado q_1 y lee el símbolo blanco desde los cabezales de la cinta de entrada y de estado, entonces escribe en la cinta de entrada cualquier símbolo de Γ que lea de la cinta de salida. También, utilizaremos la letra σ para incluir una fila en tabla por cada símbolo $a \in \Gamma - \{\#\}$, donde reemplazamos σ por a; incluso permitiremos definir el conjunto preciso con el que se quiere reemplazar σ anotando $\sigma = \{a, b, \ldots, x\}$, y además permitiremos usar otro símbolo distinto de σ para cumplir el mismo rol, en caso que sea necesario y siempre que lo indiquemos debidamente. En el caso de los diagramas de estado, se adoptan las mismas convenciones.

Ejemplo 5.9 Consideremos el siguiente ejemplo. Sea $M = (Q, \Gamma, \Lambda, \delta, s, H)$ una máquina social tal que:

$$\begin{aligned} Q &= \{q_0, q_1, h\} \\ \Gamma &= \{\#, a, b\} \\ \Lambda &= \{\lambda\} \\ s &= q_0 \\ H &= \{h\} \end{aligned}$$

Y donde la función parcial de transición δ se define por la siguiente tabla:

δ	q	$\triangleright$	$\square$	λ	q'	$\triangleright$	$\square$	λ
1	q_0	$\#$	$\#$	$\#$	h	$\#$	$\#$	$\#$
2	q_0	$\#$	$\#$	a	h	$\#$	$\#$	a
3	q_0	$\#$	$\#$	b	h	$\#$	$\#$	b
4	q_0	a	$\#$	$\#$	q_1	a	$\#$	a
5	q_0	a	$\#$	a	q_1	a	$\#$	a
6	q_0	a	$\#$	b	q_1	a	$\#$	a
7	q_0	b	$\#$	$\#$	q_1	b	$\#$	b
8	q_0	b	$\#$	a	q_1	b	$\#$	b
9	q_0	b	$\#$	b	q_1	b	$\#$	b
10	q_1	a	$\#$	a	q_0	R	$\#$	R
11	q_1	b	$\#$	b	q_0	R	$\#$	R

Primero que todo, nótese que esta tabla sigue ciertas regularidades que podríamos resumir fácilmente utilizando las convenciones de la Observación 5.8. Se ha enumerado las filas de la tabla para reconocer más fácilmente el uso de las convenciones. Por ejemplo, en las filas 1 a 3, todas las columnas distintas al nombre del canal de salida tienen el mismo valor, mientras que el valor asociado al cabezal de la cinta de salida en cada fila es igual al valor asociado a la acción a realizar sobre esa cinta. Así, las filas 1 a 3 pueden resumirse fácilmente usando la notación ρ. Trabajando de esta forma podemos reducir el número de filas, y así definir la función de transición mediante la siguiente tabla:

δ	q	$\triangleright$	$\square$	λ	q'	$\triangleright$	$\square$	λ
[1-3]	q_0	$\#$	$\#$	ρ	h	$\#$	$\#$	ρ
[4-6]	q_0	a	$\#$	$*$	q_1	a	$\#$	a
[7-9]	q_0	b	$\#$	$*$	q_1	b	$\#$	b
[10-11]	q_1	σ	$\#$	σ	q_0	R	$\#$	R

Ahora las filas han sido etiquetadas de acuerdo al intervalo de las filas originales que resumen. De esta forma, la primera fila resume las filas 1 a 3 de la tabla inicial, la segunda resume las filas 4 a 6 de la tabla inicial, y así sucesivamente. Nótese que al usar el símbolo σ para resumir las filas 10 y 11 no se considera el caso $\sigma = \#$. Finalmente, las filas de esta tabla pueden colapsarse aún más, al combinar la fila [4-6] y la fila [7-9]. Esto produce la siguiente tabla:

q	$\triangleright$	$\square$	λ	q'	$\triangleright$	$\square$	λ
q_0	$\#$	$\#$	ρ	h	$\#$	$\#$	ρ
q_0	σ	$\#$	$*$	q_1	σ	$\#$	σ
q_1	σ	$\#$	σ	q_0	R	$\#$	R

Además se presenta el diagrama de estados de la función de transición de este ejemplo, usando las convenciones señaladas anteriormente:

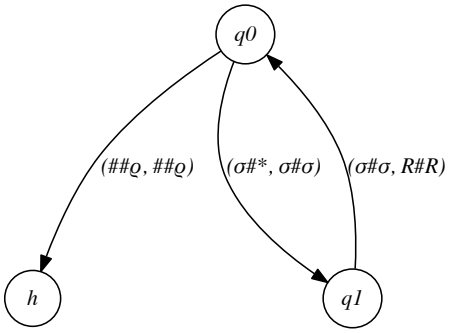


Figura 5.3: Diagrama de estados de ejemplo.

Ahora que hemos simplificado la notación de la función de transición, procederemos a considerar una cierta configuración inicial para la máquina social M, de modo que derivaremos la secuencia de micropasos que se suceden a partir de ella, ejemplificando así una computación de M. Para esto, fijemos el orden $\triangleright, \square, \lambda$, y como ejemplo tomemos la configuración inicial $C_0 = (q_0, \underline{a}bba, \underline{\#}, \underline{\#})$; entonces:

$$
\begin{array}{rllll}
 & (q_0, & \underline{a}bba, & \underline{\#}, & \underline{\#} \quad) \\
\vdash_M & (q_1, & \underline{a}bba, & \underline{\#}, & \underline{a} \quad) \\
\vdash_M & (q_0, & a\underline{b}ba, & \underline{\#}, & a\underline{\#} \quad) \\
\vdash_M & (q_1, & a\underline{b}ba, & \underline{\#}, & a\underline{b} \quad) \\
\vdash_M & (q_0, & ab\underline{b}a, & \underline{\#}, & ab\underline{\#} \quad) \\
\vdash_M & (q_1, & ab\underline{b}a, & \underline{\#}, & ab\underline{b} \quad) \\
\vdash_M & (q_0, & abb\underline{a}, & \underline{\#}, & abb\underline{\#} \quad) \\
\vdash_M & (q_1, & abb\underline{a}, & \underline{\#}, & abb\underline{a} \quad) \\
\vdash_M & (q_0, & abba\underline{\#}, & \underline{\#}, & abba\underline{\#} \quad) \\
\vdash_M & (h, & abba\underline{\#}, & \underline{\#}, & abba\underline{\#} \quad)
\end{array}
$$

Esta secuencia de configuraciones de la máquina M conforman por tanto una computación de la máquina social. Como vemos, la máquina ha copiado el contenido de su cinta de entrada en su cinta de salida. Así por ejemplo, podríamos pensar esta máquina como un dispositivo que recibe un input del entorno y lo envía de vuelta por su canal de salida.

5.2. Comportamiento persistente de máquinas sociales

Como vimos al comienzo del capítulo, toda máquina social cuenta con un canal de entrada por donde espera recibir un input desde el entorno, y uno o más canales de salida por los que espera enviar distintos outputs. Una vez que la máquina recibe un input, realiza una serie de micropasos de computación que le permiten generar los distintos outputs a ser enviados por los canales de salida. Esta actividad computacional que comienza una vez ingresado el input desde el entorno y termina cuando se ha generado cada output —siempre que la

máquina no diverja, como acotaremos más adelante— es lo que denominamos un *macropaso*. De esta forma, la máquina *interactúa con su entorno* recibiendo un input por su canal de entrada, dando un macropaso de computación, y disponiendo cada output en su canal de salida respectivo. Además, como establecimos al introducir el modelo, entre cada macropaso que la máquina realiza, ésta persiste un estado en la forma de un *string de estado*. Esto significa que la máquina puede *recordar* una palabra de su macropaso previo, y a su vez modificar esa palabra para ser utilizada en el macropaso posterior. Esto posibilita que el output generado por una máquina social pueda en principio depender del historial completo de interacciones entre ésta y su entorno. Esta suerte de *memoria* de una máquina al interactuar con su entorno fue formalizada por Goldin et al. (2004) en su modelo de máquinas persistentes de Turing. Esta sección se inspira en aquel modelo, y cuando se establezcan definiciones y resultados similares al de los autores se realizarán las observaciones pertinentes.

En la sección anterior formalizamos la noción de máquina social y modelamos su comportamiento interno; estudiamos la noción de etapa de computación bajo el concepto formal de configuración, y definimos la relación lleva en un micropaso, que da cuenta intuitivamente de cuándo una máquina social puede pasar de una configuración a otra, redefiniendo con ello el estado interno de su unidad de control y modificando posiblemente el contenido de cada una de sus cintas. Finalmente definimos una computación como una secuencia de configuraciones donde cada configuración lleva en un micropaso a la siguiente. Estas nociones que formalizamos en la sección anterior nos permitirán ahora modelar la persistencia exhibida por las máquinas sociales al interactuar con el entorno. En primer lugar introducimos el concepto de *computación persistente* que formaliza la idea intuitiva de macropaso. Luego, estudiamos métodos para definir las interacciones de una máquina social con su entorno, que resultan mucho más idóneos que el uso de la función parcial de transición. Finalmente, entramos de lleno en el terreno de la interacción persistente al introducir las nociones de *lenguaje de stream persistente* y *par de interacción persistente*, entre otros.

Como ya hemos señalado con anterioridad, una máquina social posee un canal de entrada por donde espera recibir un input, y uno o más canales de salida que usa para el envío de outputs. Vista desde el exterior, la máquina recibe un input por su canal de entrada, realiza un cómputo interno no observable por su entorno, y concluye con la disposición de un output en cada canal de salida. Sin embargo, hasta ahora no hemos especificado cómo la máquina ingresa el input desde el entorno, ni cómo la máquina dispone el contenido de sus cintas en los canales de salida. En la sección anterior hemos sido claros en señalar que cada una de las cintas de la máquina social cumple una función determinada. La cinta de entrada es usada para contener el input, la cinta de trabajo es usada para contener el string de estado, y las cintas de salida son usadas para contener cada una el output respectivo al canal de salida asociado. Lo que falta por especificar entonces son los *protocolos de inicio y término* que adoptaremos para usar las máquinas sociales; en otras palabras, debemos fijar las normas que especifiquen cuál es la configuración en la que queda una máquina social al ingresar un input desde el entorno —conteniendo además un cierto string de estado—, así como también qué configuraciones representarán el término de un macropaso y la forma en que deduciremos

el nuevo string de estado y los outputs a partir del contenido de las cintas. Introducimos a continuación estos protocolos de inicio y término con que los trabajaremos en todo el modelo:

Definición 5.10 *Sea $M = (Q, \Gamma, \Lambda, \delta, s, H)$ una máquina social, $\Sigma_0 \subseteq \Sigma - \{\#\}$, y $C = (q, l)$ la configuración actual de M. Entonces diremos que M cumple con el* protocolo de inicio de computación persistente *si para algún $u, v \in \Sigma_0^*$ se satisface:*

$$q = s, \quad l(\triangleright) = \#u\underline{\#}, \quad l(\square) = \#v\underline{\#}, \quad \forall \lambda \in \Lambda \;\; l(\lambda) = \#\underline{\#}$$

A su vez, diremos que M cumple con el protocolo de término de computación persistente *si para algún $u, v \in \Sigma_0^*$ se satisface:*

$$q \in H, \quad l(\triangleright) = \#u\underline{\#}, \quad l(\square) = \#v\underline{\#}, \quad \forall \lambda \in \Lambda \;\; l(\lambda) = \#w_\lambda\underline{\#}, \; w_\lambda \in \Sigma_0^*$$

Revisemos brevemente esta definición. Primero vemos que los protocolos de inicio y término indican que la máquina debe encontrarse en una configuración de inicio y de término, respectivamente. Luego, vemos que la cintas de entrada y de trabajo, en ambos protocolos, contienen en su primera celda un blanco #, seguido de un string que no contiene el blanco, y luego nuevamente un blanco, celda en la que también está situado el cabezal. Por su parte, las cintas de salida contienen solo dos blancos en el protocolo de inicio, y en el protocolo de término siguen el mismo patrón que las cintas de entrada y trabajo. En general, el símbolo blanco estará reservado para el uso interno de las máquinas sociales; de esta manera, los strings de input, estado y output tendrán la forma $u \in \Sigma_0 \subseteq \Sigma - \{\#\}$ y los representaremos en cada cinta como $\#u\underline{\#}$. En este sentido, ahora podemos interpretar el protocolo de inicio como una configuración inicial con un cierto string representado en la cinta de entrada, otro representado en la cinta de trabajo, y el string vacío ε representado en cada cinta de salida, mientras que el protocolo de término podemos verlo como una configuración de término con un string representado por cada cinta. Ahora que hemos definido los protocolos de inicio y término, formalizamos la idea intuitiva de macropaso a través del concepto de computación persistente.

Definición 5.11 (cf. Goldin et al., 2004, def. 6) *Sea $M = (Q, \Gamma, \Lambda, \delta, s, H)$ una máquina social, $\Sigma_0 \subseteq \Sigma - \{\#\}$, y $w_i, w \in \Sigma_0^*$. Diremos que M* computa persistentemente *el input w_i con estado inicial w si y solo si existen configuraciones $C = (q, l)$, $C' = (q', l')$ que satisfacen las siguientes condiciones:*

1. *$q = s$, $q' \in H$, y $C \vdash_M^* C'$;*
2. *$l(\triangleright) = \#w_i\underline{\#}, \quad l(\square) = \#w\underline{\#}$, y $\quad \forall \lambda \in \Lambda \;\; l(\lambda) = \#\underline{\#}$;*
3. *$l'(\triangleright) = l(\triangleright), \quad l'(\square) = \#w'\underline{\#}$, $w' \in \Sigma_0^*$, y $\quad \forall \lambda \in \Lambda \;\; l(\lambda) = \#w_\lambda\underline{\#}$, $w_\lambda \in \Sigma_0^*$.*

Si es así, diremos que w' es el estado final *de la computación, y que $w_o : \Lambda \to \Sigma_0^*, w_o(\lambda) = w_\lambda$ es el* output*. Indicaremos que M computa el input w_i con estado inicial w, estado final w' y output w_o mediante la notación:*

$$w \vDash_M w' \; [w_i, w_o] \tag{5.1}$$

También nos permitiremos fijar un orden para los nombres de canales de salida en Λ y escribir la función de output w_o en la ecuación 5.1 como la secuencia de valores asociados a esos nombres, siguiendo el orden fijado. A la secuencia de configuraciones $(C, C_1, \ldots, C_n, C')$ tal que $C \vdash C_1 \vdash \ldots \vdash C_n \vdash C'$ la llamaremos computación persistente*. En adelante, usaremos el término p-computación como alternativa al de computación persistente.*

Revisemos brevemente esta definición. En primer lugar, la condición *1* establece que para que M compute persistentemente un cierto input con cierto estado inicial, debe haber una computación en el sentido de la Definición 5.7, tal que una configuración de inicio lleve en cero o más micropasos a una configuración de parada. Esto representa la idea intuitiva y bastante obvia de que un macropaso es una sucesión finita de micropasos con alguna propiedad particular. Luego, la condición *2* establece que la configuración de inicio debe cumplir con el protocolo de inicio, que el string representado en la cinta de entrada debe ser el input, y que el string representado en la cinta de trabajo debe ser el estado inicial. Finalmente, la condición *3* establece que la configuración de parada debe cumplir el protocolo de término, que el input debe seguir siendo representado en la cinta de entrada, y además establece cómo interpretar los strings representados en las cintas de trabajo y salida, a saber, el string representado en la cinta de trabajo será el string de estado final, y el string representado en cada cinta de salida será el output asignado a cada canal de salida. Esto termina de especificar completamente cuál es el rol que cumple cada cinta dentro de la definición de máquina social.

Ahora estamos en condiciones de introducir la noción de persistencia en el modelo de máquinas sociales. Esto lo logramos haciendo que la máquina realice secuencias de computaciones persistentes donde el estado final de cada computación persistente corresponda al estado inicial de la siguiente. Supongamos que una máquina social M recibe un input w_i del entorno, y que se encuentra con string de estado w; entonces, la máquina procederá a realizar una p-computación que producirá en su cinta de trabajo un nuevo string de estado w' y, por cada cinta de salida λ, un output w_λ. Recordando de la Definición 5.11 que el output es una función w_o tal que $w_o(\lambda) = w_\lambda$, en una interacción con su entorno M recibe de parte de éste un input w_i, realiza una p-computación, y envía de vuelta al entorno un output w_o. Más tarde, llamaremos al par (w_i, w_o) *par de interacción persistente*; por ahora, lo importante de destacar es que M quedó con w' en su cinta de trabajo, y la próxima vez que realice una p-computación, su cinta de trabajo mantendrá intacto ese string; esa es la esencia de la noción de persistencia. Esto significa que al recibir un nuevo input w_i^2 del entorno, la máquina social M realizará una p-computación con estado inicial w', produciendo un output w_o^2 y un nuevo string de estado w'' para el que se repite la explicación anterior.

En la Figura 5.4 se ilustra la noción de computación persistente para una máquina social de una sola cinta de salida. La figura exhibe el contenido de cada una de las cintas de esta máquina social en cuatro etapas de computación distintas. En la esquina superior izquierda, se ilustra la máquina social con estado w una vez que ha ingresado el input w_i del entorno. Luego, la flecha celeste a su derecha indica que se lleva a cabo una p-computación, lo que termina con las cintas dispuestas según se ilustra en la esquina superior derecha. Notemos lo siguiente. Primero, la cinta de entrada tiene el mismo contenido al inicio y término de la computación; esto sugiere que, al ser la cinta de entrada aquella donde se dispone el input, la máquina debiera utilizar esta cinta sólo como una cinta de lectura. En esta versión del modelo, no obstante, no se ha incluido esta restricción de *solo lectura* dentro del formalismo.

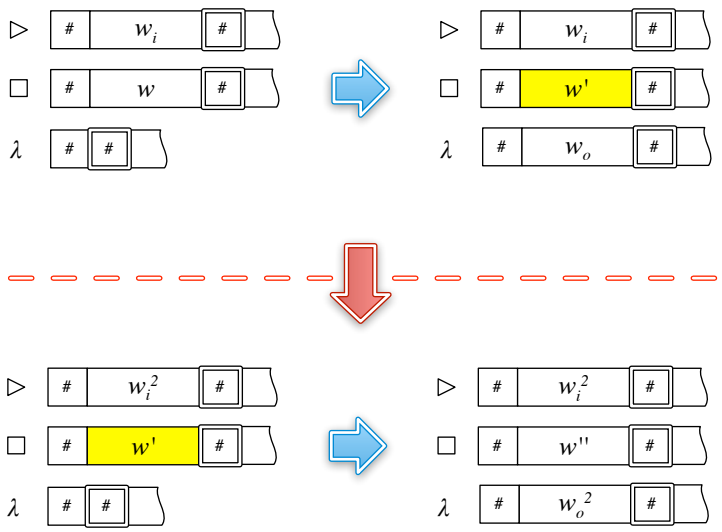


Figura 5.4: Persistencia en la computación.

Segundo, la cinta de salida representa al inicio de la p-computación el string vacío ε, y al término de ésta el output a enviar por el canal de salida. Esto da cuenta de la idea intuitiva de que al comienzo de la p-computación los canales de salida están vacíos; también sugiere que las cintas de salida debieran poder ser de *solo escritura*. No obstante, tampoco incluimos esta restricción en el formalismo. Ahora bien, la siguiente etapa de computación se ilustra en la esquina de abajo a la izquierda. Esta etapa sucede luego de que la máquina ha dispuesto su output en su canal de salida, y el entorno le ha enviado un nuevo input w_i^2 a su canal de entrada, con lo que comienza una nueva p-computación. Esta transición de una p-computación a otra está señalizada por la flecha roja vertical al centro de la figura. Así, la mitad superior de la figura ilustra una p-computación, y la mitad inferior ilustra la siguiente. La persistencia entre estas dos p-computaciones se da entonces en la medida que la máquina social utiliza el estado final w' de la primera p-computación como estado inicial de la segunda. Esto se ilustra destacando en amarillo el string de estado compartido por las p-computaciones. A continuación, revisamos en un ejemplo la noción de computación persistente que acabamos de introducir.

Ejemplo 5.12 En este ejemplo se introducirá la máquina social M_{Latch}. El comportamiento de esta máquina consiste, básicamente, en recibir una palabra en binario del entorno y enviar por su único canal de salida el primer bit de la palabra anterior que el entorno le había enviado. De otra forma, el estado final que persiste en esta máquina a una nueva p-computación, es el primer bit del input inicial que recibió, y el output de cada p-computación es el propio estado inicial. En el caso que el string de estado de la máquina es ε, ésta responde por defecto con un 0, y en el caso que el input sea vacío, entonces el estado final será también vacío. Formalmente, definimos esta máquina social como $M_{\text{Latch}} = (Q, \Gamma, \Lambda, \delta, s, H)$, de modo que:

$$
\begin{aligned}
Q &= \{q_0, q_1, q_2, q_3, q_4, q_5, h_1, h_2\} \\
\Gamma &= \{\#, 0, 1\} \\
\Lambda &= \{\lambda\} \\
s &= q_0 \\
H &= \{h_1, h_2\}
\end{aligned}
$$

Y donde la función parcial de transición δ se define mediante el diagrama de estados ilustrado en la Figura 5.5.

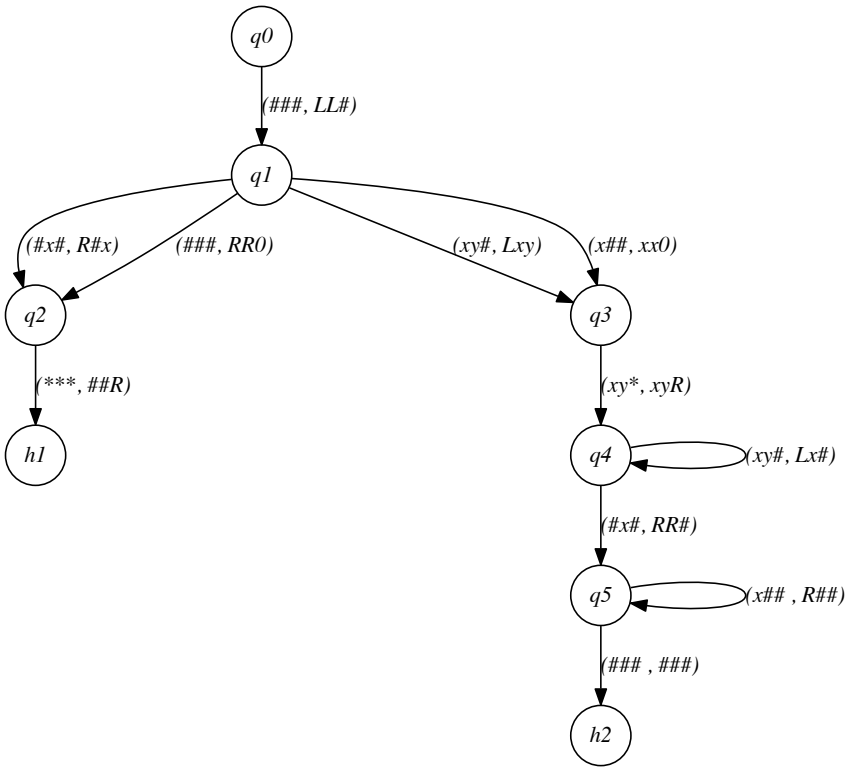


Figura 5.5: Diagrama de estados de M_{Latch}.

Para construir este diagrama de estados se hizo uso de las convenciones adoptadas en la Observación 5.8 de la sección anterior, donde hemos ocupado x, y como variables para recorrer el conjunto $\{0, 1\}$ —de acuerdo a la notación σ que allí incorporamos—. Revisando rápidamente este diagrama, y asumiendo el protocolo de inicio de p-computación, vemos que la máquina comienza desplazando a la izquierda los cabezales de las cintas de entrada y trabajo. Si luego el cabezal de la cinta de entrada lee blanco, entonces la máquina recibió ε del entorno; por tanto, si su string de estado era vacío responde con un 0, y si no, lo escribe en la cinta de salida, y almacena el input ε como string de estado de una forma muy peculiar, a saber, sobreescribiendo un blanco # en la celda que mantenía el string de estado que, recordemos, era de solo un bit. Este primer comportamiento esta descrito por la rama izquierda del diagrama. La rama derecha, por su parte, se ocupa del caso en que el input no es vacío. Aquí la máquina parte escribiendo el output correcto en la cinta de salida, luego examina la cinta de entrada hasta encontrar el primer símbolo del input, y lo escribe en la celda que contiene el único bit en que consiste el string de estado. Luego la máquina se dispone para cumplir el protocolo de término.

Ahora que hemos visto cómo se define esta máquina, presentamos algunos ejemplos de computaciones persistentes que M_{Latch} puede llevar a cabo.

$$\varepsilon \vDash 1\ [100101, 0] \tag{5.2a}$$

$$1 \vDash 0\ [001100, 1] \tag{5.2b}$$

$$0 \vDash 0\ [011, 0] \tag{5.2c}$$

$$0 \vDash 1\ [1000001, 0] \tag{5.2d}$$
$$1 \vDash \varepsilon\ [\varepsilon, 1] \tag{5.2e}$$
$$\varepsilon \vDash 0\ [00, 0] \tag{5.2f}$$

Partamos por la ecuación (5.2a). Esta ecuación, por ejemplo, permite modelar el comportamiento inicial que puede exhibir la máquina M_{Latch} : al ser la primera vez que interactúa con su entorno, no ha almacenado un string de persistencia, y por lo tanto éste puede suponerse vacío. Al recibir el string 100101, guarda el primer bit 1 como estado, y envía por defecto un 0 por su canal de salida. Cuando la máquina ya almacena un bit como string de estado, lo envía como output al entorno a la vez que configura su nuevo estado como el primer bit del input que recibe. Este comportamiento podemos apreciarlo en las ecuaciones (5.2b), (5.2c) y (5.2d). En la ecuación (5.2e) se presenta una computación persistente donde el input es vacío, y por tanto la máquina configura su string de estado como vacío. La ecuación (5.2f) retrata nuevamente el caso en que la máquina tiene estado inicial vacío; lo que puede verse nuevamente como el inicio de una secuencia de computaciones persistentes, o también como el resultado de interactuar con el entorno luego de la p-computación establecida en la ecuación anterior (5.2e). De hecho, como el lector ya habrá supuesto, la secuencia de ecuaciones que se presentó refleja una posible secuencia de interacciones entre M_{Latch} y su entorno, donde M_{Latch} computa el output a enviar al entorno a partir del input que recibe y un estado que persiste entre cada computación. Esto sucede porque en la secuencia de p-computaciones, el estado final de cada p-computación es igual al estado inicial de la p-computación siguiente. La intuición nos dice que no tendría sentido haber introducido la noción de p-computación, si no exigiéramos igualar los estados finales e iniciales de dos p-computaciones sucesivas. De hecho, e informalmente hablando, es este *enganche* el que abre la posibilidad de todo un comportamiento computacional persistente que va más allá de la noción inicial de p-computación. En efecto, una vez que formalicemos esta idea de *enganche*, contaremos con las nociones de persistencia que este ejemplo busca reflejar. Sin embargo, antes debemos facilitarnos el trabajo de definir máquinas sociales, utilizando para ello solo la información de las p-computaciones que queremos que éstas exhiban, y omitiendo los detalles de estados y transiciones que hacen más complejo el asunto.

Observación 5.13 En el Ejemplo 5.12, cuando se explicó cómo operaba la máquina social de acuerdo a la información contenida en el diagrama de estados, se destacó el siguiente hecho. Si el string de estado no era vacío, pero la máquina recibía el input vacío ε, entonces ésta guardaba ε como string de estado escribiendo en la cinta de trabajo un blanco # a la izquierda de la celda donde parte el cabezal. Nótese que en el caso que el string de estado fuera un bit, esto cumple satisfactoriamente con el comportamiento que pronosticamos para M_{Latch}. Sin embargo, si el string de estado inicial hubiera sido una palabra wa con $w \in \Sigma_0^*$, $a \in \Sigma_0$, entonces el string de estado final hubiese sido w. Esto significa que M_{Latch} no siempre almacena un bit como estado. Sin embargo, toda p-computación de M_{Latch} con estado inicial de largo 0 o 1 se comporta como esperamos, y en efecto, construimos M_{Latch} para esos casos. Así, podemos concluir dos cosas. La primera, que aún sabiendo cuál es el comportamiento que queremos que tenga una máquina social, resulta muy difícil definir la función de transición para considerar cada uno de los casos posibles, incluso considerando su parcialidad y las notaciones y convenciones que hemos adoptado. La segunda conclusión, es que resulta bastante conveniente contar con una forma de definir máquinas sociales en términos de sus p-computaciones. Esto significa que debemos buscar una manera de asegurar que con sólo decir cómo queremos que compute una máquina, podemos encontrarla —si es que existe— y utilizarla.

Como hemos visto ya en el Ejemplo 5.12 y en la Observación 5.13, resulta muy conveniente contar con una forma de especificar máquinas sociales solamente en base al comportamiento que queremos que exhiban. En lo que sigue exploramos estas posibilidades. Con este propósito en mente, lo primero que haremos será ocuparnos de un escenario que hasta el momento no hemos considerado en el modelo, a saber, que un macropaso no finalice por tratarse de una computación divergente. En este caso, la máquina nunca llegaría a una configuración de parada, y por tanto tampoco habría string de estado final ni outputs que disponer en los canales de salida. Más aún ¿Cómo podría una máquina social volver a interactuar con su entorno si no ha completado la interacción previa? En este sentido, seguimos el enfoque adoptado por Goldin et al. (2004), quienes modelan la situación asumiendo que la máquina entra en un escenario especial de divergencia del cual no vuelve a salir. No importa cuántas veces reciba un input desde el entorno, si la máquina comienza su computación en este escenario de divergencia, entonces termina la computación también en el escenario de divergencia y dispone en sus canales de salida un output especial para indicar divergencia. Específicamente, la divergencia se representa asignando al string de estado un estado especial s_{div}, y a la función de output una función especial μ que asocia el output especial de divergencia μ a cada elemento de su dominio[1]. De este modo, si la máquina diverge, de ahi en adelante su string de estado será siempre s_{div} y su función de output será siempre μ. De este modo, cualquier agente del entorno que intente interactuar con la máquina reconocerá que está se encuentra en el escenario de divergencia. Formalizamos esta noción a continuación:

Definición 5.14 (cf. Goldin et al., 2004, def. 6) *Sea $M = (Q, \Gamma, \Lambda, \delta, s, H)$ una máquina social, $\Sigma_0 \subseteq \Sigma - \{\#\}$, y $w_i, w \in \Sigma_0^*$. Diremos que M* diverge *ante el input w_i con estado inicial w si y solo si dada la configuración $C = (s, l)$, con $l(\triangleright) = \#w_i\underline{\#}$, $l(\square) = \#w\underline{\#}$, y $\forall\lambda \in \Lambda \;\; l(\lambda) = \#\underline{\#}$ ésta no se detiene, es decir, se satisface:*

$$\nexists C' = (q', l') \in \mathcal{C}_M, \;\; q' \in H \wedge C \vdash_M^* C'$$

Si es así, escribiremos:

$$w \vDash s_{div}\ [w_i, \mu] \tag{5.3}$$

Donde $s_{div}, \mu \notin \Sigma^*$; s_{div} es un estado especial, el estado de divergencia*, que satisface para cualquier input w_i:*

$$s_{div} \vDash s_{div}\ [w_i, \mu] \tag{5.4}$$

Y donde μ es un símbolo especial para denotar la función de output en la divergencia.

A esta altura del modelo, ya sabemos exactamente qué significa que una máquina social M compute persistentemente un input w_i con estado inicial w, produciendo un estado final w' y un output w_o; así también, sabemos exactamente qué significa que M diverja ante el input w_i y con estado inicial w. Sin embargo, estas nociones que hemos formalizado son, por decirlo de algún modo, definiciones bastante *puntuales*, ya que operan sobre un solo input y estado a la vez. Dicho de otra forma, si pudiéramos definir una máquina social a través del conjunto de p-computaciones que esperamos que exhiba, con las nociones que hemos incorporado hasta

[1]Nótese que se está recargando el significado asociado al signo μ, por lo que seremos bastante claros al usarlo especificando si nos referimos a la función misma o al valor de la función.

ahora en el modelo tendríamos que especificar una por una todas las p-computaciones. En este sentido, resulta sumamente conveniente poder definir el comportamiento de una máquina social en términos de una función total o parcial que a cada par (w_i, w) de input y estado inicial, le asocie un par (w', w_o) de estado final y función de output. Con este propósito, incorporamos en el modelo la noción de *computación persistente de funciones*, con la cual podremos describir el comportamiento ideal de una máquina social mediante una función f, asegurando la existencia de esa máquina en la medida que f sea persistentemente computable. Se introduce esta definición a continuación:

Definición 5.15 *Sea Σ_0 un alfabeto que no contiene el símbolo blanco #. Sea f una función total o parcial de $\Sigma_0^* \times \Sigma_0^*$ en $\Sigma_0^* \times (\Sigma_0^*)^n$. Sea $M = (Q, \Gamma, \Lambda, \delta, s, H)$ una máquina social con $|\Lambda| = n$. Definimos las siguientes nociones de computación persistente:*

1. *Si f es una función total, diremos que M* computa persistentemente *f si y solo si $\Sigma_0 \in \Sigma$ y existe una indización de las cintas de salida,* index : $I_n \to \Lambda$ *tal que para cada $(w_i, w) \in \Sigma_0^* \times \Sigma_0^*$ se cumple:*

$$f(w_i, w) = (w', \overline{w}) \Rightarrow \ w \vDash_M w' \ [w_i, w_o], \ w_o(\lambda_k) = \pi_k(\overline{w}) \ \ \forall k \in I_n \tag{5.5}$$

   Si existe tal M diremos que f (total) es persistentemente computable.

2. *Si f es una función parcial, diremos que M* computa persistentemente *f si y solo si $\Sigma_0 \in \Sigma$ y existe una indización de las cintas de salida,* index : $I_n \to \Lambda$ *tal que para cada $(w_i, w) \in \Sigma_0^* \times \Sigma_0^*$ se cumple:*

   si f está definida en (w_i, w), se satisface 5.5
   si f no está definida en $(w_i, w) \Rightarrow M$ diverge ante el input w_i con estado inicial w

   Si existe tal M diremos que f (parcial) es persistentemente computable.

3. *Si f es una función parcial, diremos que M* computa persistente y parcialmente *f si y solo si $\Sigma_0 \in \Sigma$ y existe una indización de las cintas de salida,* index : $I_n \to \Lambda$ *tal que para cada $(w_i, w) \in \Sigma_0^* \times \Sigma_0^*$ se cumple:*

   si f está definida en (w_i, w), se satisface 5.5
   si f no está definida en $(w_i, w) \Rightarrow M$ puede o no divergir ante el input w_i con estado inicial w

   Si existe tal M diremos que f es parcialmente y persistentemente computable.

Para definir las máquinas sociales de forma más sucinta, y así evitar especificar cada uno de los objetos que la componen, adoptaremos en este modelo dos enfoques. En el primero, definiremos las máquinas sociales mediante el uso de la *notación modular*. Al igual que en el caso de máquinas de Turing, esta notación nos permitirá conectar máquinas entre sí, y con ello construir nuevas máquinas a partir de la composición de otras preexistentes. Este es el primer enfoque que exploramos. En el segundo enfoque, en cambio, derivamos un resultado crucial que relaciona el concepto de función Turing computable —descrito en el capítulo anterior, Definición 4.10— con el de función persistentemente computable. Con este resultado, podremos asegurar que si el comportamiento que deseamos que tenga una máquina social se puede representar como una función Turing computable, entonces tal máquina social existe.

Así contaremos con toda la teoría ya conocida de Turing computabilidad de funciones a nuestro favor para especificar máquinas sociales. Pero primero, introducimos la especificación de máquinas sociales usando notación modular.

La notación modular de máquinas sociales consiste sencillamente en una adaptación de la notación modular de máquinas de Turing. Primero que todo, al definir una máquina social M usando notación modular, supondremos que todas las máquinas sociales que se utilizan como módulos para componer M comparten exactamente el mismo alfabeto básico de comunicación Γ, y el mismo conjunto de nombres de canales de salida Λ. Es decir, el alfabeto sobre el que opera toda máquina social que participa de la notación es exactamente el mismo, a saber, $\Sigma = \Gamma \cup \Lambda$. Junto con esto, también debemos fijar un ordenamiento del conjunto de identificadores de cintas Λ_0. La notación modular funciona de la siguiente forma. Primero, establecemos un conjunto de máquinas sociales que serán los nodos en la notación. Luego, establecemos un conjunto de arcos dirigidos para conectar nodos, los que deben estar etiquetados con una secuencia de símbolos en Σ, uno por cada identificador de cinta, bajo el orden preestablecido. Finalmente, establecemos cuál es el nodo inicial M_s de la notación. Hecho esto, interpretaremos el diagrama que habremos elaborado como una máquina social M que computa de la siguiente forma. M comienza simulando M_s hasta que M_s llega a una configuración de parada. Luego, M revisa el contenido bajo sus cabezales. Si existe algún arco que parta en M_s y tenga como etiqueta una secuencia de símbolos que, de acuerdo al orden preestablecido, se corresponda con la información de los cabezales, entonces M sigue el arco hasta el nodo siguiente y repite el proceso. En el caso que esto no ocurra, M detiene su computación. Ahora bien, corresponde hacer algunas observaciones. Primero, se ha preferido desde un punto de vista metodológico omitir la definición formal de la notación modular; esto porque no se considera un mayor aporte a la presentación del modelo de máquinas sociales. Segundo, las máquinas sociales pueden *copiarse* fácilmente renombrando cada uno de sus estados; así, nos permitiremos anotar una misma máquina muchas veces subentendiendo que en realidad son muchas copias de la misma máquina. Tercero, usaremos la notación modular de forma determinista, esto es, nunca nos veremos en la situación que podemos transitar por más de un arco. Cuarto, hasta ahora hemos supuesto que las máquinas que emula M en cada nodo por el que pasa tienen computaciones convergentes. En el caso que M emule una máquina cuya computación diverja, entonces M también diverge. En quinto y último lugar, nótese que para introducir la notación modular no hemos hecho uso de las nociones de persistencia o p-computación; es decir, éste es un método *genérico* para expresar máquinas en términos de otras. A continuación definimos las *acciones básicas* de la notación modular. Estas acciones básicas son los primeros nodos que tendremos a disposición para construir nuevas máquinas, las que a su vez estarán disponibles para seguir definiendo máquinas sociales modularmente.

Definición 5.16 *Las* acciones básicas *de la notación modular de máquinas sociales de definen de la siguiente manera*[2]*:*

1. *Por cada $\lambda_0 \in \Lambda_0$, se define la acción básica* Left_{λ_0} *como la máquina que mueve a la*

[2]Por razones lógicas, se ha omitido en cada tabla los identificadores de las cintas que no se alteran en un micropaso; usando las convenciones de la Observación 5.8, podemos representar esto definiendo para cada identificador $\lambda_k \in \Lambda_0, \lambda_k \neq \lambda_0$ una variable de reemplazo σ_k que recorra todos los símbolos en Σ, y ubicándola bajo las columnas de las secciones izquierda y derecha asociadas al identificador λ_k.

izquierda el cabezal de la cinta asociada a λ_0*. Esto es,* $\text{Left}_{\lambda_0} = (Q, \Gamma, \Lambda, \delta, s, H)$ *donde* $Q = \{s, h\}$*,* $H = \{h\}$*, y* δ *está definida por la siguiente tabla:*

q	λ_0	q'	λ_0
s	$*$	h	L

2. *Por cada* $\lambda_0 \in \Lambda_0$*, se define la acción básica* Right_{λ_0} *como la máquina que mueve a la derecha el cabezal de la cinta asociada a* λ_0*. Esto es,* $\text{Right}_{\lambda_0} = (Q, \Gamma, \Lambda, \delta, s, H)$ *donde* $Q = \{s, h\}$*,* $H = \{h\}$*, y* δ *está definida por la siguiente tabla:*

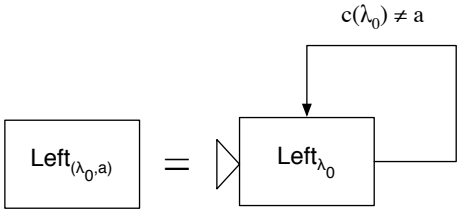
q	λ_0	q'	λ_0
s	$*$	h	R

3. *Por cada* $a \in \Sigma$ *y* $\lambda_0 \in \Lambda$ *se define la acción básica* $\text{Write}_{a/\lambda_0}$ *como la máquina que escribe* a *en cabezal de la cinta asociada a* λ_0*. Esto es,* $\text{Write}_{a/\lambda_0} = (Q, \Gamma, \Lambda, \delta, s, H)$ *donde* $Q = \{s, h\}$*,* $H = \{h\}$*, y* δ *está definida por la siguiente tabla:*

q	λ_0	q'	λ_0
s	$*$	h	a

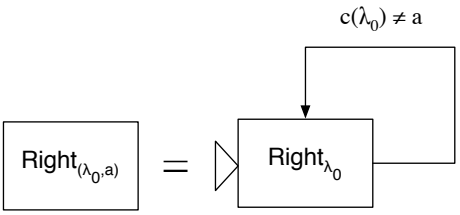
A continuación ejemplificamos el uso de la notación modular utilizando las acciones básicas recién descritas.

Ejemplo 5.17 En este ejemplo definiremos tres máquinas sociales utilizando para ello las acciones básicas de la notación modular. Primero, especificamos la máquina $\text{Left}_{(\lambda_0,a)}$. Esta máquina comienza emulando Left_{λ_0} por lo que al inicio mueve el cabezal de la cinta λ_0 una posición a la izquierda. Luego, examina la celda bajo el cabezal; si ésta contiene el símbolo a, la máquina termina; si no, la celda contiene un símbolo distinto, por lo que vuelve al nodo inicial con lo se repite el ciclo. De esta forma la máquina $\text{Left}_{(\lambda_0,a)}$ mueve el cabezal de la cinta λ_0 a la izquierda hasta encontrar el símbolo a, o bien se cuelga. En la siguiente figura se define modularmente esta máquina:

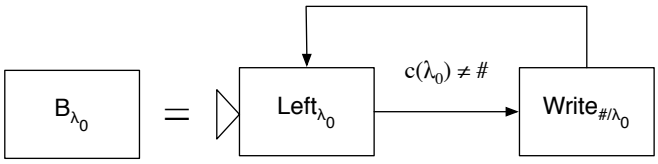


Examinemos brevemente esta figura. Primero, en el miembro izquierdo de la igualdad dibujamos la representación modular de la máquina que queremos especificar. A la derecha, dibujamos los nodos y los arcos. Para indicar el nodo inicial, dibujamos un triángulo que apunte al este y lo posicionamos junto al borde izquierdo del nodo inicial. Nótese que en esta definición, hemos etiquetado la arista del único arco como $c(\lambda_0) \neq a$. En términos simples, la etiqueta significa que el cabezal de la cinta λ_0 debe leer un símbolo distinto de a. En general dibujaremos arcos con etiquetas del tipo $c(\lambda) \neq x$ para representar el conjunto de arcos que, conectando los mismos nodos y en la misma dirección, tienen como etiqueta cada una de las posibles secuencias de símbolos, tales que el símbolo x no aparece en la posición que corresponde al identificador λ, según el orden de identificadores preestablecido.

Ahora definimos la máquina $\text{Right}_{(\lambda_0,a)}$ de la siguiente forma:



En base a lo que discutimos anteriormente, resulta sencillo entender como funciona esta máquina. $\text{Right}_{(\lambda_0,a)}$ desplaza el cabezal de la cinta λ_0 hasta encontrar el símbolo a; en caso contrario, la computación diverge. El último ejemplo que revisaremos se especifica en la siguiente figura:



Esta máquina comienza moviendo el cabezal de la cinta λ_0 a la izquierda, y mientras no lea un blanco # en ese cabezal, escribe un blanco y pasa directamente al nodo inicial, con lo que se repite el ciclo. En caso que el cabezal de la cinta λ_0 contenga un blanco, la máquina se detiene. Nótese que en esta figura, hemos incluido un arco sin etiqueta. Esto significará que sin importar lo que lo que lean los cabezales, la máquina pasa directamente al nodo de llegada. Obsérvese, no obstante, que esto implica que el nodo de llegada es en efecto simulado por la máquina. Esta última máquina de ejemplo B_{λ_0} que hemos introducido tiene un comportamiento que será muy útil al definir otras máquinas usando notación modular. Debido a que esta máquina se mueve a la izquierda y escribe un blanco # hasta leer otro, si el contenido de la cinta λ_0 es de la forma $\#w\underline{\#}$, con $w \in \Sigma - \{\#\}$, entonces al emularse B_{λ_0}, la cinta quedará con contenido $\underline{\#}$. Esto es, B_{λ_0} es la máquina que borra la cinta λ_0, lo que será muy útil al incorporarla como módulo para definir otras máquinas.

Ejemplo 5.18 En este ejemplo construiremos una máquina copiadora utilizando la notación modular y las máquinas sociales que ya definimos modularmente en el ejemplo anterior. Dados dos identificadores de cinta λ_j, λ_k, una máquina social copiadora $C_{\lambda_j\lambda_k}$ es aquella que al partir de una configuración de inicio (s, l) donde $l(\lambda_j) = \#u\underline{\#}$, y $l(\lambda_k) = \#v\underline{\#}$, lleva en uno o más micropasos a una configuración de término (h, l') donde $l'(\underline{\lambda_j}) = \#u\underline{\#}$ y $l'(\underline{\lambda_k}) = \#v\#u\underline{\#}$, sin modificar el resto de sus cintas. Es decir, copia u a la derecha del contenido de la cinta λ_k. Especificamos esta máquina en la Figura 5.6. Como se ilustra en la figura, la máquina comienza moviendo el cabezal de la cinta λ_j a la izquierda hasta encontrar un blanco, luego mueve el cabezal de la cinta λ_k a la derecha, y entra en un ciclo tal que mueve a la derecha el cabezal de la cinta λ_j y, si lee un blanco, entonces ya no queda nada por copiar y para, sino, escribe en la cinta λ_k lo que lee en la cinta λ_j, mueve el cabezal de la cinta λ_k a la derecha y repite el ciclo. De esta forma se van copiando uno a uno los símbolos de la cinta λ_j en la cinta λ_k. Nótese que hemos dibujado un arco con la etiqueta $c(\lambda_j) = \sigma \neq \#$, y luego hemos ocupado ese símbolo σ como parte del nombre del nodo al cual ir. En general, usaremos esta notación cuando hallamos parametrizado una máquina por un cierto conjunto de símbolos. De esta forma, el siguiente nodo al que pasa la máquina depende del símbolo en el cabezal.

Ya exploramos el primer enfoque a incorporar para facilitar la definición de máquinas sociales. Ahora examinamos el segundo. En este enfoque, hacemos uso de la nociones de computabilidad persistente para asegurar la existencia de máquinas sociales. En específico,

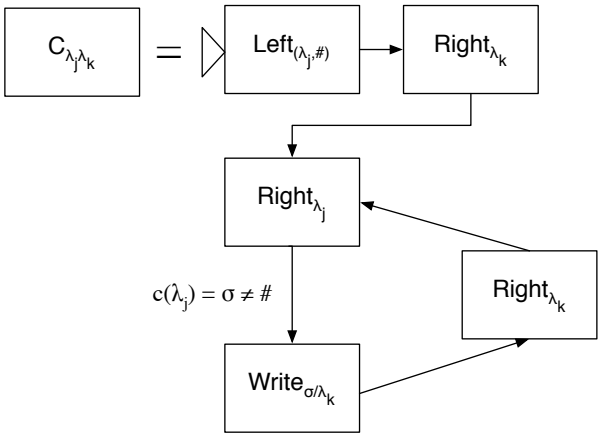


Figura 5.6: Máquina social copiadora.

supongamos que queremos definir una máquina social tal que p-compute de acuerdo a lo que nosotros queramos. Supongamos, también, que podemos describir por medio de una función f esas p-computaciones que queremos sean las que exactamente lleve a cabo la máquina. De esta forma, f asocia a cada par de input y estado inicial (w_i, w), un estado final y función de output (w', w_o) respectivo. Ahora bien, por la Definición 5.15 sabemos que si f es persistentemente computable, entonces existe una máquina social que lleva a cabo exactamente las p-computaciones que deseamos. En realidad, esa *es* la propia definición de computabilidad persistente. Entonces ¿Cómo podemos determinar si la función f es peristentemente computable? En este segundo enfoque ofrecemos una respuesta a esta pregunta que nos permitirá contar a nuestro favor con toda la teoría ya conocida de Turing computabilidad. Presentamos esa respuesta en la forma de la siguiente proposición.

Proposición 5.19 *Sea Σ_0 un alfabeto que no contiene el símbolo blanco $\#$. Sea f una función total definida como $f : \Sigma_0^* \times \Sigma_0^* \to \Sigma_0^* \times (\Sigma_0^*)^n$ y g una función parcial definida como $g : \Sigma_0^* \times \Sigma_0^* \nrightarrow \Sigma_0^* \times (\Sigma_0^*)^n$. Entonces se tiene que:*

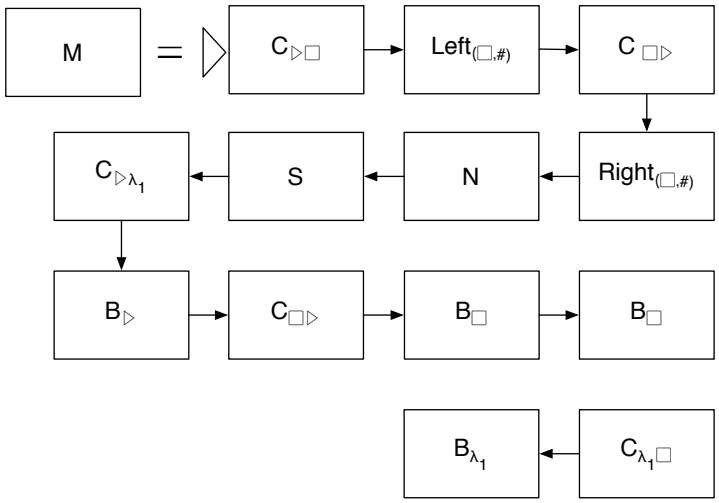
1. *Si f (total) es Turing computable, entonces f (total) es p-computable.*
2. *Si g (parcial) es Turing computable, entonces g (parcial) es p-computable.*
3. *Si g es parcialmente Turing computable, entonces g es parcialmente p-computable.*

Demostración. Demostaremos sólo el primer caso pues, como veremos, el resto de los casos son análogos. Primero damos una idea de la demostración. Sea T la máquina de Turing que computa f. Construiremos una máquina social N que solo trabajará sobre su cinta de entrada simulando en ella el comportamiento de T. Esto es, en cada micropaso N modificará su cinta de entrada de la misma forma en que T modificaría su única cinta al dar un paso de computación de Turing. De esta forma, N podrá *computar* f en su cinta de entrada de la misma forma que lo hace T. Es decir, N no cumplirá el protocolo de inicio y término de computación persistente, sino que computará f de acuerdo al formato que se establece en la Definición 4.10 sobre Turing computabilidad. Luego, para p-computar f construiremos una máquina M que pasará desde el protocolo inicial de p-computación al protocolo que usa N para computar f, correremos N, y luego llevaremos el resultado al protocolo de término de p-computación. M será la máquina que buscamos.

Sea entonces $T = (Q, \Sigma, \delta, s, H)$ la máquina de Turing que computa f. Definimos N como $N = (Q, \Gamma, \Lambda, \tau, s, H)$, donde Γ y Λ son alfabetos escogidos de cualquier forma tal que satisfagan la propia Definición 5.1 de máquinas sociales y las condiciones $\Sigma_0 \subseteq \Gamma \cup \Lambda = \Sigma$ y $|\Lambda| = n$. Además fijamos una indización de Λ, index : $I_n \to \Lambda$. Definimos la función parcial de transición como $\tau : (Q - H) \times (\Lambda_0 \to \Sigma) \nrightarrow Q \times (\Lambda_0 \to (\Sigma \cup \{L, R\}))$ de modo que:

$$\tau(q, c) = (p, c[a/\triangleright]) \qquad \text{donde } \ \delta(q, x) = (p, a),\ x = c(\triangleright)$$

Aquí la notación $c[a/\triangleright]$ denota la función que resulta de reemplazar con a el valor que c asociaba a la preimagen $\triangleright$, como se detalla en la Definición 4.4. Nótese que la ecuación anterior significa que para cada par (q, c), τ pasa al mismo estado que δ, las cintas distintas a $\triangleright$ mantienen el contenido de sus celdas, y la cinta $\triangleright$, que lee $x = c(\triangleright)$, realiza la misma acción a que T realiza al ver x. Continuando con la demostración, definimos la máquina M como $M = (K, \Gamma, \Lambda, \tau', s', H')$, donde los estados y transiciones se definen usando notación modular en el siguiente diagrama:



Y donde S es una máquina auxiliar que copia el $(k+1)$-ésimo string sobre Σ_0 representado en la cinta de entrada, a la cinta de salida λ_k, para luego borrar ese string de la cinta de input y repetir el ciclo, lo que va desde $k = n$ a $k = 1$. S se define modularmente como sigue. Sea P_k la máquina compuesta por $C_{\triangleright,\lambda_k}$ y $B_\triangleright$, con una flecha desde $C_{\triangleright,\lambda_k}$ hasta $B_\triangleright$, y que parte en $C_{\triangleright,\lambda_k}$. Entonces S se define como la máquina compuesta por P_k para cada $k \in I_n$, con una flecha desde P_i hasta P_{i-1} para cada $1 < i \leq n$, y que parte en P_n.

M funciona de la siguiente forma. Inicialmente copia el input en su cinta de trabajo, su string de estado en la cinta de entrada, y posiciona el cabezal de la cinta de trabajo a la derecha de la copia que se hizo del input. Nótese que, hasta este punto, si M parte desde el protocolo de inicio de p-computación entonces no diverge. Luego corre N, con lo que la cinta de entrada queda con $k+1$ strings sobre Σ_0 rodeados por blancos como establece la Definición 4.10. Después continúa con S, que distribuye los últimos k strings sobre Σ_0 en las k cintas de salida. Nótese que, hasta este punto, si N no ha divergido, entonces M tampoco lo ha hecho. Luego de correr S, queda solo un string en la cinta de entrada, el estado final. El resto de M se preocupa de dejar este string en la cinta de estado, y de dejar en la cinta de entrada la copia que se hizo del input al empezar M —para hacer esto, M utiliza temporalmente la primera cinta de salida—. Si N no divergió, entonces M termina su computación.

Es claro que como f es total, entonces N siempre converge, y por lo tanto M también. Sea $f(w_i, w) = (w', \overline{w})$. Por el párrafo anterior, si M parte del protocolo inicial de p-computación con input w_i y

estado w, entonces posiciona correctamente w_i y w para que N compute f, y luego distribuye correctamente $\overline{w}$ y w' en la cintas de salida y estado. Además deja la cinta de entrada con el input original. Por lo tanto, cuando M se detiene, cumple con el protocolo de término de p-computación con estado final w' y $w_o(\lambda_k) = \pi_k(\overline{w})\ \ \forall k \in I_n$, con lo que se cumple:

$$f(w_i, w) = (w', \overline{w}) \Rightarrow\ w \vDash_M w'\ [w_i, w_o],\ w_o(\lambda_k) = \pi_k(\overline{w})\ \ \forall k \in I_n$$

Esto demuestra el primer punto. En los otros casos, remplazamos T por la máquina de Turing que corresponda, digamos T'. Entonces tendremos que M se detendrá si y solo si T' lo hace, y en caso que lo haga M seguirá cumpliendo con la formula anterior. Así, cuando T' compute g, M p-computará g, y cuando T' compute parcialmente g, M p-computará parcialmente g, con lo que se tiene los resultados *2* y *3*. □

Ejemplo 5.20 En el Ejemplo 5.12 introducimos la máquina social M_{Latch} que recibía inputs en binario y almacenaba como nuevo string de estado el primer bit de su input, a la vez que retornaba como output por su único canal de salida su string de estado inicial. Cuando su string de estado inicial era vacío, entonces retornaba por defecto un 0. Para definir la máquina del ejemplo, utilizamos el diagrama de estados exhibido en la Figura 5.5, y dimos algunos ejemplos de computaciones persistentes entre los que se encontraban:

$$\begin{aligned}
&\varepsilon \vDash 1\ [100101, 0]\\
&1 \vDash 0\ [001100, 1]\\
&0 \vDash 0\ [011, 0]\\
&0 \vDash 1\ [1000001, 0]
\end{aligned}$$

Sin embargo, inmediatamente después de la presentación del Ejemplo 5.12, tuvimos que, a través de la Observación 5.13, advertir al lector que este comportamiento se daba sólo cuando la máquina M_{Latch} partía desde un string de estado inicial de largo 0 o 1. El diagrama de estados especificaba comportamientos imprevistos bajo estados iniciales distintos a los que presupusimos. Entendimos lo complejo que resultaba definir cada posible caso de la función de transición usando tablas y diagramas de estados aún con las notaciones y convenciones que adoptamos para facilitar la escritura. Y, en vez de dedicarnos a reescribir perfectamente la función parcial de transición de M_{Latch}, concluímos que debíamos desarrollar métodos más idóneos para definir máquinas sociales. En este ejemplo definimos correctamente la máquina M_{Latch} haciendo uso de la proposición anterior. Nótese la simpleza de la definición.

Sea $\Sigma_0 = \{0, 1\}$ un alfabeto y f la función parcial $f : \Sigma_0^* \times \Sigma_0^* \nrightarrow \Sigma_0^* \times \Sigma_0^*$ definida según:

$$f(w_i, w) = \begin{cases}
(\varepsilon, 0) & \text{si } |w_i| = |w| = 0\\
(w_i[1], 0) & \text{si } |w_i| \neq 0 \ \text{ y } \ |w| = 0\\
(w_i[1], w) & \text{si } |w_i| \neq 0 \ \text{ y } \ |w| = 1\\
\text{no definido} & \text{si } |w| \geq 2
\end{cases}$$

Claramente f (parcial) es Turing computable. Entonces, por la Proposición 5.19, f (parcial) es persistentemente computable, y así definimos la máquina social M_{Latch} como aquella que la p-computa.

Observación 5.21 Nótese que en la demostración de la Proposición 5.19 se especifica que los alfabetos Γ y Λ de la máquina social M que se usa para lograr la demostración pueden ser, en estricto rigor, escogidos de cualquier forma tal que satisfagan la definición de máquinas sociales y

las condiciones $\Sigma_0 \subseteq \Gamma \cup \Lambda = \Sigma$ y $|\Lambda| = n$. De esta forma, por un concepto de completitud en la definición de M_{Latch} establecemos:

$$\Gamma = \{\#, 0, 1\}$$
$$\Lambda = \{\lambda\}$$

En general, haremos este tipo de especificaciones siempre que sea necesario.

Corolario 5.22 *Sea $\mathbb{L}$ un lenguaje de programación Turing completo —como Java o C—. Entonces podemos usar $\mathbb{L}$ para definir máquinas sociales en términos de sus p-computaciones.*

Demostración. Por la Proposición 5.19, sabemos que dada una función Turing computable f, existe una máquina social M que p-computa según f. Por otro lado, dado un lenguaje de programación $\mathbb{L}$ Turing completo, sabemos que podemos definir f por una expresión E en $\mathbb{L}$. Luego, esa expresión E define f que a su vez define las p-computaciones de M. □

Observación 5.23 Nótese que el corolario anterior no señala que toda máquina social M pueda definirse por sus p-computaciones usando un lenguaje de programación Turing completo $\mathbb{L}$. Esto, no obstante, es cierto. Más aún, el recíproco de cada una de las cláusulas de la Proposición 5.19 es verdadero también. Sin embargo, no exploramos esos resultados en este trabajo.

Ejemplo 5.24 En este ejemplo usamos el Corolario 5.22 para introducir la máquina social grabadora M_G. Esta máquina simula el comportamiento de una grabadora que tiene tres operaciones: reproducir, borrar y grabar. Para ello, la máquina recibe del entorno tres inputs posibles: $PLAY$, $ERASE$ y $RECORDsome\_entry$; donde $some\_entry$ es lo que se desea grabar. Ante cualquier otro input la máquina diverge. La máquina tendrá un solo canal de salida $\Lambda = \{\lambda\}$ y su alfabeto básico de comunicación Γ será el alfabeto español. Así, definimos la máquina grabadora M_G a través de la siguiente expresión en pseudocódigo:

```
1:  function GRABADORA(command, content)
2:      if command = "PLAY" then
3:          return command, content
4:      else if command = "ERASE" then
5:          return null, "OK"
6:      else if startswith(command, "RECORD") then
7:          n ← length("RECORD")
8:          entry ← substring(command, n)          ▷ substring desde posición n en adelante
9:          newcontent ← concat(content, entry)
10:         return newcontent, "OK"
11:     else
12:         loop
```

Revisemos brevemente los p-cómputos que define este pseudocódigo. Cuando M_G p-computa con input $PLAY$, su output corresponde al string de estado, el que no se altera durante la p-computación. Cuando M_G p-computa con input $ERASE$, entonces su string de estado final es ε —lo que se ha expresado usando el valor nulo en el pseudocódigo anterior— y su output es OK. Cuando M_G p-computa un input que comienza con el string $RECORD$, entonces M_G considera el substring que viene después de $RECORD$ en el string de input, y lo concatena al string de estado inicial. Ese

string resultante corresponde al estado final, mientras que el output correspondiente es OK. Ante cualquier otro input M_G diverge. Terminamos presentando algunos ejemplos de computaciones persistentes que M_G puede llevar a cabo.

ε	$\models \varepsilon$	$[PLAY, \varepsilon]$
ε	$\models \sqcup$sudamérica	$[RECORD \sqcup \text{sudamérica}, OK]$
$\sqcup$ sudamérica	$\models \sqcup$sudamérica $\sqcup$ es	$[RECORD \sqcup \text{es}, OK]$
$\sqcup$ sudamérica $\sqcup$ es	$\models \sqcup$sudamérica $\sqcup$ es $\sqcup$ bella	$[RECORD \sqcup \text{bella}, OK]$
$\sqcup$ sudamérica $\sqcup$ es $\sqcup$ bella	$\models \sqcup$sudamérica $\sqcup$ es $\sqcup$ bella	$[PLAY, \sqcup\text{sudamérica} \sqcup \text{es} \sqcup \text{bella}]$
$\sqcup$ sudamérica $\sqcup$ es $\sqcup$ bella	$\models \varepsilon$	$[ERASE, OK]$
ε	$\models s_{div}$	$[trash, \mu]$

Ahora que ya hemos perfeccionado nuestros métodos para definir máquinas sociales, dedicamos el resto de la sección a estudiar en mayor detalle las nociones de *interacción* y *persistencia*. Como se ha reiterado, una máquina social es un dispositivo computacional que cuenta con un canal de entrada y uno o más canales de salida. Internamente, la máquina se compone de una cinta de entrada y una o más cintas de salida, que se asocian correspondientemente a los canales de entrada y salida. Además, la máquina cuenta con una cinta de trabajo, que forma parte del comportamiento interno de la máquina, y por lo tanto es inobservable desde el entorno. Al comienzo de esta sección establecimos cómo interpretaríamos el ingreso de un input a la máquina mediante el establecimiento de un protocolo de inicio. También indicamos como deduciríamos los outputs a disponer en los canales de salida a partir del contenido de las cintas de salida mediante el protocolo de término. Definimos la noción formal de computación persistente y luego, en el Ejemplo 5.12, cuando dimos algunos ejemplos de p-computaciones, comprendimos que la definición de p-computación tiene sentido sólo si logramos establecer de alguna forma que el estado inicial de una p-computación sea el estado final de la p-computación anterior. Es decir, debemos exigir lo que entonces llamamos informalmente el *enganche* entre p-computaciones. Hacemos eso a continuación.

Veamos primero la idea. Sea M una máquina social. Para modelar el hecho que M mantiene un cierto string de estado w —que no debe contener el símbolo blanco—, supondremos que el contenido de su cinta de trabajo es $\#w\underline{\#}$. Cuando el entorno envíe un input w_i a M, ésta lo ingresará de acuerdo al protocolo de inicio y p-computará con input w_i y el string de estado que contenga en la cinta de trabajo, en este caso w. Asumiendo que la computación converge con estado final w' y función de output w_o, la máquina dispondrá el output en cada canal de salida según corresponda, y w' será el nuevo string de estado a persistir. Esto significa que desde ese momento hasta que M vuelva a recibir un input del entorno, el contenido de la cinta de trabajo será $\#w'\underline{\#}$. Así, cuando la máquina vuelva a recibir un input w_i^2 desde el entorno, realizará una computación persistente con input w_i^2 y estado inicial w' de forma que, asumiendo convergencia, generará un estado final w'' y output w_o^2. Como resultado, M tendrá en su cinta de trabajo el contenido $\#w''\underline{\#}$, y así el ciclo se volverá a repetir en adelante. Cada par de la forma (w_i, w_o) representa una interacción con el entorno,

y por lo tanto lo llamaremos *par de interacción*. Así, la interacción de una máquina con el entorno se puede pensar como un par de interacción tras otro de forma que el contenido de la cinta de trabajo de la máquina persiste entre cada par. Estas ideas se formalizan bajo la forma de dos nociones distintas, aunque relacionadas. La primera es la de *stream de interacción*, concepto introducido por Goldin et al. (2004) en su modelo de máquinas persistentes de Turing; intuitivamente un stream de interacción corresponde a un stream de la forma $s = (a, s')$ tal que a es un par de interacción y s' es un stream de interacción *posible* dado el estado que ahora persiste la máquina. La segunda noción es la de *secuencia de interacción*, que corresponde a una secuencia de pares de interacción donde se cumple el *enganche* del que se hablaba. A continuación vamos a definir la primera de estas nociones. Se recomienda al lector revisar la Sección 4.3 sobre streams del capítulo anterior. En ella se tratan los aspectos mínimos requeridos para lo que resta de sección. La noción de stream de interacción se da en términos del conjunto donde estos se definen, conjunto que se conoce con el nombre de *lenguaje de stream persistente* de una máquina social. Este *lenguaje* que habla la máquina especifica cada uno de sus comportamientos posibles. Introducimos esta definición a continuación.

Definición 5.25 (cf. Goldin et al., 2004, def. 7) *Sea* $M = (Q, \Gamma, \Lambda, \delta, s, H)$ *una máquina social,* $\Sigma_0 \subseteq \Sigma - \{\#\}$*, y* $w \in \Sigma_0^*$*. Sea* $S = \Sigma_0^* \times ((\Lambda \to \Sigma_0^*) \cup \{\mu\})$*. Se define el* Lenguaje de Stream Persistente de M con estado w *de la siguiente forma:*

$$PSL(M, w) = \{\langle (w_i, w_o), \sigma' \rangle \in S^\infty : \exists w' \in \Sigma_0^* \cup \{s_{div}\}, \\ w \vDash_M w' \ [w_i, w_o] \ \wedge \sigma' \in PSL(M, w')\}$$

El Lenguaje de Stream Persistente de M*,* $PSL(M)$*, se define como* $PSL(M) = PSL(M, \varepsilon)$*. Si dos máquinas* M_1 *y* M_2 *son tales que* $PSL(M_1) = PSL(M_2)$*, entonces se dirá que son PSL-equivalentes, y se anotará* $M_1 =_{PSL} M_2$*.*

Revisemos brevemente esta definición. Primero, el lenguaje de stream persistente de una máquina social M con estado w es un conjunto de streams. Segundo, la fórmula que define este conjunto establece que la primera componente de cada stream es un par de interacción que resulta de una computación persistente $w \vDash_M w' \ [w_i, w_o]$, mientras que la segunda es un elemento del lenguaje de stream persistente donde las p-computaciones se dan con estado inicial w'. Esto es exactamente lo que queremos modelar. En tercer lugar, nótese que esta definición considera la divergencia, es decir, se pueden describir interacciones con máquinas en escenario de divergencia, las que, por lo tanto, dispondrán en cada interacción el símbolo especial del valor del output en divergencia, μ, por sus canales de salida. Por último, nótese que el lenguaje de stream persistente de una máquina corresponde a su lenguaje de stream persistente con estado ε. Esto formaliza la idea intuitiva del comportamiento *desde el inicio* de una máquina.

Ejemplo 5.26 A continuación damos algunos ejemplos de la noción anterior haciendo uso de las máquinas de ejemplo que ya conocemos, a saber, la máquina M_{Latch} del Ejemplo 5.20 y la máquina

M_G del Ejemplo 5.24. Por ejemplo, sabemos que M_{Latch} realiza las siguientes p-computaciones:

$$\begin{aligned}
&\varepsilon \vDash 1\ [100101, 0]\\
&1 \vDash 0\ [001100, 1]\\
&0 \vDash 0\ [011, 0]\\
&0 \vDash 1\ [1000001, 0]\\
&1 \vDash \varepsilon\ [\varepsilon, 1]\\
&\varepsilon \vDash 0\ [00, 0]
\end{aligned}$$

Cuando presentamos esto en el Ejemplo 5.12 no teníamos un medio formal para indicar que estas ecuaciones respondían efectivamente a la idea de persistencia entre computaciones. Ahora podemos hacerlo usando la noción de lenguaje de stream persistente. De esta forma podemos escribir:

$$\langle (100101, 0), \langle (001100, 1), \langle (011, 0), \langle (1000001, 0), \langle (\varepsilon, 1), \langle (00, 0), s\rangle\rangle\rangle\rangle\rangle\rangle \in PSL(M_{\text{Latch}})$$

Donde $s \in PSL(M, 0)$. Con esto logramos dar cuenta formalmente que estas interacciones sí se suceden unas a otras persistiendo el string de estado. Sobre la notación, algunas veces nos permitiremos simplificar el uso de los paréntesis y omitir la información del último stream; por ejemplo la formula anterior quedaría:

$$\langle (100101, 0), (001100, 1), (011, 0), (1000001, 0), (\varepsilon, 1), (00, 0), \ldots\rangle \in PSL(M_{\text{Latch}})$$

Dos ejemplos más de streams de interacción de M_{Latch} son:

$$\begin{aligned}
&\langle (1110, \mu), (0011, \mu), (110, \mu), (0111111, \mu), \ldots\rangle \in PSL(M_{\text{Latch}}, 0001)\\
&\langle (1001, 1), (0, 1), (\varepsilon, 0), (111, 0), \ldots\rangle \in PSL(M_{\text{Latch}}, 1)
\end{aligned}$$

Nótese que M_{Latch} diverge cada vez que comienza una p-computación con un string de estado inicial de largo mayor o igual a 2. Por esto, la primera de las ecuaciones anteriores señala que la máquina responde con μ en cada interacción, lo que es propio de una máquina que ha entrado en divergencia. En efecto, puede demostrarse que para todo $u \in \{0,1\}^*$ tal que $|u| \geq 2$:

$$PSL(M_{\text{Latch}}, u) = (\{0,1\}^* \times \{\mu\})^\infty$$

Aunque no lo demostraremos aquí, es fácil ver por qué se satisface la ecuación anterior. Cuando M_{Latch} comienza una p-comutación con un string de estado inicial de largo mayor o igual a 2, la máquina diverge. Esto quiere decir que la primera interacción produce output μ y hace que la máquina entre en estado de divergencia, por lo que seguirá produciendo output μ en cada interacción futura. Así, todo par de interacción consiste en un input $w_i \in \{0,1\}^*$ y output $w_o = \mu$, lo que hace que los streams de interacción adopten la forma de la ecuación anterior. Finalizamos entregando un ejemplo de stream de interacción de la máquina social grabadora M_G:

$$\begin{aligned}
&\langle (PLAY, \varepsilon),\\
&\ (RECORD\sqcup\text{sudamérica}, OK),\\
&\ (RECORD\sqcup\text{es}, OK),\\
&\ (RECORD\sqcup\text{bella}, OK),\\
&\ (PLAY, \sqcup\text{sudamérica}\sqcup\text{es}\sqcup\text{bella}),\\
&\ \ldots\rangle \in PSL(M_G)
\end{aligned}$$

Ahora que tenemos la definición formal de lenguaje de stream persistente, podemos formalizar también la noción de par de interacción. Hacemos esto a continuación[3]:

Definición 5.27 *Sea* $M = (Q, \Gamma, \Lambda, \delta, s, H)$ *una máquina social,* $\Sigma_0 \subseteq \Sigma - \{\#\}$ *y* $S = \Sigma_0^* \times ((\Lambda \to \Sigma_0^*) \cup \{\mu\})$. *Diremos que* $(w_i, w_o) \in S$ *es un* par de interacción *de* M *si y solo si existe* $\sigma \in PSL(M)$ *tal que* $(w_i, w_o) \prec \sigma$. *También extendemos la noción diciendo que* $(w_i, w_o) \in S$ *es un par de interacción de* M *al partir del estado* w *si y solo si existe* $\sigma \in PSL(M, w)$ *tal que* $(w_i, w_o) \prec \sigma$.

Ejemplo 5.28 A continuación ejemplificamos la definición anterior. Por ejemplo la máquina M_{Latch} cuenta con los pares de interacción $(100101, 0)$, $(001100, 1)$, y $(011, 0)$ entre otros. También podemos asegurar que nunca un par de la forma (w_i, μ) con $w_i \in \{0,1\}^*$ será un par de interacción de M_{Latch}, porque como vimos desde un inicio, si esta máquina parte con string de estado inicial de largo 0 o 1 entonces su string de estado final también será de largo 0 o 1, y por lo tanto al arrancarla desde el estado vacío nunca tendrá una computación divergente. Por otro lado $(1, \mu)$ es un par de interacción de M_{Latch} al partir del estado $w = 00$. En el caso de la máquina grabadora M_G, algunos pares de interacción son $(PLAY, \sqcup\text{un}\sqcup\text{mensaje}\sqcup\text{especial})$, $(ERASE, OK)$, $(trash, \mu)$, $(RECORD\sqcup\text{mi}\sqcup\text{canción}, OK)$. Así vemos que el concepto de par interacción da cuenta efectivamente de la posibilidad de interacción que la máquina tiene con el entorno.

Al introducir esta última parte de la sección señalamos que podíamos aproximarnos de dos modos a la descripción de las ideas de interacción y persistencia que discutíamos. La primera era bajo la noción de streams de interacción, que resultaron ser los elementos de un lenguaje de stream persistente. Ahora desarrollamos la segunda aproximación que corresponde al concepto de *secuencia de interacción*. Contando ya con la noción de stream de interacción, esta tarea se simplifica bastante. Definiremos una secuencia de interacción como un *prefijo* de un stream de interacción. Disponemos las próximas dos definiciones para formalizar esta noción.

Definición 5.29 *Sea* A^∞ *el conjunto de streams sobre un conjunto* A. *Se define el operador de* prefijo, $\text{pref}_A : \mathbb{P} \times A^\infty \to A^*$, *con* $k \in \mathbb{P}$, *y* $\sigma = (a, \sigma') \in A^\infty$ *como:*

$$\text{pref}_A(k, \sigma) = \begin{cases} a & \text{si } k = 1 \\ \langle a, \text{pref}_A(k-1, \sigma') \rangle & \text{si no} \end{cases}$$

Donde $\langle\ ,\ \rangle$ *es el operador de concatenación de tuplas descrito en la Definición 4.1. Se omitirá el subíndice del operador cuando éste se sobre entienda.*

Definición 5.30 *Sea* $M = (Q, \Gamma, \Lambda, \delta, s, H)$ *una máquina social y* $\Sigma_0 \subseteq \Sigma - \{\#\}$. *Sea* $S = \Sigma_0^* \times ((\Lambda \to \Sigma_0^*) \cup \{\mu\})$. *Diremos que* $s \in S^*$ *es una* secuencia de interacción *de* M *si y solo si*

$$\exists w \in \Sigma_0^* \cup \{s_{div}\}\ \exists \sigma \in PSL(M, w)\ \exists k \in \mathbb{P} \quad \text{pref}(k, \sigma) = s$$

Si es así, diremos que $k \in \mathbb{P}$ *es el* largo *de la secuencia, y que* w *es su* estado inicial*; en el caso que* $w = \varepsilon$, *diremos que* s *es una* secuencia inicial de interacción.

[3]La notación $a \prec \sigma$ se introdujo en la Definición 4.14 del capítulo anterior.

Observación 5.31 Nótese que al igual que en el caso de los lenguajes de stream persistentes, una secuencia de interacción de una máquina social siempre está bien definida, aún cuando una de esas interacciones signifiquen una computación divergente. Esto se debe, como ya hemos señalado, al hecho de haber introducido el escenario de divergencia con su estado especial de divergencia s_{div} y output especial de divergencia μ.

Ejemplo 5.32 Basándonos en los casos vistos en el Ejemplo 5.28 tenemos los siguientes ejemplos de secuencias de interacción:

$$((100101, 0), (001100, 1), (011, 0), (1000001, 0), (\varepsilon, 1), (00, 0))$$
$$((001100, 1), (011, 0), (1000001, 0), (\varepsilon, 1), (00, 0))$$
$$((RECORD\sqcup\text{bella}, OK), (PLAY, \sqcup\text{sudamérica}\sqcup\text{es}\sqcup\text{bella}))$$
$$((RECORD\sqcup\text{bella}, OK), (PLAY, \sqcup\text{ella}\sqcup\text{es}\sqcup\text{bella}))$$

En lo que resta de sección, estudiaremos una clase particular de lenguajes llamados *lenguajes de stream amnésicos*, también introducidos por Goldin et al. (2004) en su modelo de máquinas de Turing persistentes. Este tipo de lenguaje nos permitirá darnos cuenta de la real importancia que significa el haber incorporado la noción de persistencia en nuestro modelo, a la vez que nos llevará a validar categóricamente la incorporación de la cinta de trabajo interna en la composición de la máquina social. En simple, en un lenguaje de stream amnésico los stream de interacción no pueden persistir su estado, es decir, comienzan cada nueva interacción con ε en su cinta de trabajo. Basándonos en estos lenguajes, introduciremos la noción de máquina social amnésica, la que nos permitirá evaluar cuán gravitante es la persistencia para el comportamiento de una máquina social. Se establece esta definición a continuación:

Definición 5.33 (cf. Goldin et al., 2004, def. 37) *Sea $M = (Q, \Gamma, \Lambda, \delta, s, H)$ una máquina social y $\Sigma_0 \subseteq \Sigma - \{\#\}$. Sea $S = \Sigma_0^* \times ((\Lambda \to \Sigma_0^*) \cup \{\mu\})$. Se define el* Lenguaje de Stream Amnésico de M *de la siguiente forma:*

$$ASL(M) = \{\langle (w_i, w_o), \sigma' \rangle \in S^\infty : \exists w' \in \Sigma_0^* \cup \{s_{div}\}, \\ \varepsilon \vDash_M w' \, [w_i, w_o] \ \wedge \sigma' \in ASL(M)\}$$

Si dos máquinas M_1 y M_2 son tales que $ASL(M_1) = ASL(M_2)$, entonces se dirá que son ASL-equivalentes, y se anotará $M_1 =_{ASL} M_2$. Además, definimos $\mathbb{ASL} = \{ASL(M) : M$ es una máquina social $\}$ y diremos que M es una máquina amnésica si $PSL(M) \in \mathbb{ASL}$.

Ejemplo 5.34 A continuación ejemplificamos estas nociones con las máquinas sociales M_{Latch} y M_G que hemos estado usando. Respecto de la máquina M_{Latch}, tenemos que:

$$ASL(M_{\text{Latch}}) = \{\langle (w_i, 0), s \rangle : w_i \in \{0,1\}^* \ \wedge \ s \in ASL(M_{\text{Latch}})\}$$

Lo anterior se debe a que esta máquina responde con 0 cuando su string de estado inicial es vacío, es decir, toda p-computación con string de estado inicial ε es de la forma $\varepsilon \vDash w' \, [w_i, 0]$. Ahora bien, por la propia Definición 5.33 de lenguaje de stream amnésico, $ASL(M_{\text{Latch}})$ es el conjunto de todos los streams que se pueden escribir de esta forma, y por lo tanto, reescribiendo esta ecuación tenemos que es el conjunto más grande que satisface:

$$ASL(M_{\text{Latch}}) = (\{0,1\}^* \times \{0\}) \ \times \ ASL(M_{\text{Latch}})$$

Luego, por la Definición 4.11 de streams se tiene que:

$$ASL(M_{\text{Latch}}) = (\{0,1\}^* \times \{0\})^\infty$$

En el caso de la máquina grabadora M_G, un ejemplo de stream en $ASL(M_G)$ es:

$$\langle (PLAY, \varepsilon), (RECORD\sqcup\text{una}\sqcup\text{frase}, OK), (PLAY, \varepsilon), \ldots \rangle$$

Lo que permite ver con claridad el efecto que puede tener el *olvidar* el string estado entre p-computaciones.

Como ya sabemos, entre cada interacción con su entorno una máquina social mantiene en su cinta de trabajo un string de estado. Al recibir un input desde el entorno, éste es ingresado en la cinta de entrada según el protocolo de inicio que establecimos. Luego, la máquina lleva a cabo una computación persistente, y dispone el contenido de sus cintas de salida en los respectivos canales de salida de acuerdo al protocolo de término. Ahora el contenido de la cinta de trabajo contiene el nuevo string de estado y la máquina lo almacenará hasta volver a interactuar con su entorno. Este mecanismo de persistencia del que dotamos a las máquinas sociales les permite *recordar* información de sus interacciones anteriores y por lo tanto mostrar un comportamiento externo distinto ante dos envíos del mismo input por parte del entorno. Ahora bien, la noción de máquina amnésica nos permite evaluar cuánto descansa el comportamiento de una máquina social en la posibilidad de persistir un estado entre interacciones. Gracias a la Definición 5.33 podemos ahora expresar formalmente la idea de que el comportamiento externo que exhibe un máquina puede no requerir de la persistencia. Aún cuando una máquina amnésica en efecto persiste un estado entre cada interacción, que puede o no ser vacío, su lenguaje de stream persistente, y por lo tanto cada stream de interacción, es emulable por una máquina a la que no se le permite recordar su estado entre interacciones. Esto tiene una notable consecuencia sobre cómo esta máquina interactúa con su entorno: su output está completamente determinado por el input que recibe. En lo que sigue, presentamos dos proposiciones que nos permitirán formalizar esta idea. Finalmente, veremos con un ejemplo el drástico impacto que esto puede tener en la expresividad de la máquina.

Proposición 5.35 (cf. Goldin et al., 2004, lem. 39) *Sea $M = (Q, \Gamma, \Lambda, \delta, s, H)$ una máquina social, $\Sigma_0 \subseteq \Sigma - \{\#\}$ y $S = \Sigma_0^* \times ((\Lambda \to \Sigma_0^*) \cup \{\mu\})$. Sea $L(M)$ definido como sigue:*

$$L(M) = \{(w_i, w_o) \in S : \exists w' \in \Sigma_0^* \cup \{s_{div}\},\ \varepsilon \vDash_M w'\ [w_i, w_o]\}$$

Entonces $ASL(M) = L(M)^\infty$.

Demostración. Debemos mostrar que $ASL(M) = L(M) \times ASL(M)$, y luego por la propia definición de $ASL(M)$ este conjunto será el más grande tal que satisfaga esa ecuación. Veamos primero que $ASL(M) \subseteq L(M) \times ASL(M)$. Se tiene por definición que $(w_i, w_o) \in S$ y $\exists w' \in \Sigma_0^* \cup \{s_{div}\}$ $\varepsilon \vDash_M w'\ [w_i, w_o]$, con lo que tenemos $(w_i, w_o) \in L(M)$. Además, $\sigma' \in ASL(M)$ por definición, con lo que se tiene tiene la primera inclusión. Veamos ahora la segunda inclusión, $L(M) \times ASL(M) \subseteq ASL(M)$. Sea $\sigma = \langle (w_i, w_o), \sigma' \rangle \in L(M) \times ASL(M)$. Por definición tenemos que $(w_i, w_o) \in S$ y $\exists w' \in \Sigma_0^* \cup \{s_{div}\}$, $\varepsilon \vDash_M w'\ [w_i, w_o]$. Además como $\sigma' \in ASL(M)$ se cumple con la definición de lenguaje de stream amnésico, lo que concluye la demostración. □

Proposición 5.36 *Sea* $M = (Q, \Gamma, \Lambda, \delta, s, H)$ *una máquina social,* $\Sigma_0 \subseteq \Sigma - \{\#\}$ *y* $S = \Sigma_0^* \times ((\Lambda \to \Sigma_0^*) \cup \{\mu\})$. *Sea* $I_p(M)$ *el conjunto de todos los pares de interacción persistentes posibles de* M*, es decir,* $I_p(M) = \{(w_i, w_o) \in S : \exists \sigma \in PSL(M)\ (w_i, w_o) \prec \sigma\}$. *Entonces, si* M *es una máquina amnésica y* $(w_i, w_o) \in I_p(M)$*, se tiene que:*

$$\forall (u, v) \in I_p(M) \quad u = w_i \Rightarrow v = w_o$$

De forma análoga, si definimos $I_a(M) = \{(w_i, w_o) \in S : \exists \sigma \in ASL(M)\ (w_i, w_o) \prec \sigma\}$*, entonces si* $(w_i, w_o) \in I_a(M)$*, se tiene que:*

$$\forall (u, v) \in I_a(M) \quad u = w_i \Rightarrow v = w_o$$

Demostración. Supongamos primero que M es una máquina amnésica. Entonces existe N tal que $ASL(N) = PSL(M)$. Por la Proposición 5.35 vista recientemente, $ASL(N) = L(N)^\infty$, para cierto $L(N) \subseteq S$. Sea $(u, v) \in I_p(M)$ y supongamos por contradicción que existe $(u, w) \in I_p(M), v \neq w$. Entonces existen streams σ y σ' en $L(N)^\infty$ tales que $(u, v) \prec \sigma$ y $(u, w) \prec \sigma'$. Entonces, por definición de stream se tiene que $(u, w), (u, v) \in L(N)$, lo que implica que $\varepsilon \vDash_N x_1[u, v]$ a la vez que $\varepsilon \vDash_N x_2[u, w]$ con $v \neq w$; pero esto es una contradicción debido a que las máquinas sociales son deterministas, ante el mismo input y estado inicial generan el mismo output y estado final. Para el segundo caso, sea M es una máquina social cualquiera. Nuevamente, por la Proposición 5.35 tenemos que $ASL(M) = L(M)^\infty$ para cierto $L(M) \subseteq S$. El resto de la demostración es análoga al caso anterior. □

De esta forma, la Proposición 5.36 comprueba la idea intuitiva que detallábamos más arriba, a saber, el hecho de que una máquina amnésica genera siempre el mismo output ante el mismo input. Esto significa que externamente el comportamiento de la máquina está totalmente determinado por el input específico que recibe del entorno, sin importar interacciones previas. Pero entonces, por ejemplo, se tiene que estas máquinas no poseen la capacidad de modificar su comportamiento en base al input de sus usuarios, algo tan básico como sustancial en el comportamiento de la mayoría de las máquinas sociales reales; esto da cuenta de la gran importancia que tiene el haber incluido la noción de persistencia en el modelo de máquinas sociales. Finalizamos la sección con el siguiente ejemplo.

Ejemplo 5.37 En este ejemplo veremos cómo toda una clase de máquinas sociales de la vida real son imposibles de modelar a través de máquina amnésicas. Este el caso de las máquinas basadas en interacciones de la forma consulta-respuesta y que además reciben *feedback* de sus usuarios. Este feedback califica de alguna forma el nivel de la respuesta, y es usado por la máquina para mejorar a su vez el nivel de las respuestas futuras. Un ejemplo de este tipo de máquinas es el buscador de Google —haciendo las simplificaciones pertinentes, claro—. En este caso, un usuario realiza una consulta, que concluye con una lista ordenada de resultados entregados por Google. Al elegir un resultado por sobre otro en la lista, el usuario está dando un feedback a Google, el que será considerado, entre otros, para generar la próxima respuesta a la misma consulta. Dicho lo anterior, vamos a simplificar los detalles y suponer para efectos del presente ejemplo que este tipo de máquinas recibe su feedback en la forma de un string $fb \in \{\text{yes}, \text{no}\}$. Sea $\Sigma_0 \subseteq \Sigma - \{\#\}$ y $S = \Sigma_0^* \times (\Lambda \to \Sigma_0^*)$. Definimos un *lenguaje orientado por feedback* L_{fb} como:

$$L_{fb} \subseteq \{\langle (q, r), \langle (fb, \varepsilon), \sigma \rangle \rangle : \ (q, r) \in S \ \wedge \ fb = \{\text{yes}, \text{no}\} \ \wedge \ \sigma \in L_{fb}\}$$

Donde hemos abusado de la notación escribiendo ε para representar la función que asigna ε a cada identificador de cinta de salida. De esta forma, podemos describir la clase de máquinas de las que hablábamos diciendo que su lenguaje de stream persistente es un lenguaje orientado por feedback. Como vemos de la fórmula anterior, los streams de interacción consisten de un par consulta-respuesta seguido de un par feedback-vacío y así sucesivamente. Nótese además que hemos supuesto que no hay computaciones divergentes.

Ahora bien, la Proposición 5.36 nos muestra cuán restrictivo es el hecho de que una máquina social sea amnésica. En efecto, supongamos que M es una máquina amnésica y que $PSL(M)$ es un lenguaje orientado por feedback. Supongamos además que la siguiente es una secuencia de interacción de M:

$$((q_1, r_1), (\text{no}, \varepsilon))$$

Entonces, sin importar lo que pase, r_1 seguirá siendo el output ante q_1, independientemente de la respuesta negativa que entregó el usuario. En otras palabras, aún cuando la máquina ingresa el feedback del usuario, y aún cuando ésta persiste strings de estados posiblemente no vacíos, la máquina nunca varía su respuesta ante el mismo input. Esto muestra claramente cómo con esta clase de máquinas es imposible modelar un comportamiento tan básico de las máquinas sociales como es el modificarse por el input de los usuarios, lo que también muestra cuán gravitante es el uso efectivo de la persistencia.

5.3. Comportamiento externo de máquinas sociales

En esta última parte del modelo nos dedicamos a formalizar la interacción entre las máquinas sociales y las personas. La primera parte de nuestro modelo estuvo destinada a formalizar la noción de máquina social y describir el comportamiento interno de ésta entendiendo cómo operaban las cintas y la unidad de control. En la segunda parte del modelo incorporamos la noción de persistencia y estudiamos cómo la máquina almacenaba su string de estado entre interacciones sucesivas con su entorno. Para describir estas interacciones asumimos que la máquina recibía un input en su canal de entrada y disponía en sus canales de salida el output respectivo. Sin embargo, no distinguimos emisores ni receptores en ese entorno, ni tampoco detallamos qué sucedía en el momento anterior de haber recibido el input, ni en el momento posterior de haber generado el output. De alguna forma el entorno enviaba y recibía mensajes hacia y desde la máquina, la que entonces no nos preocupó especificar mayormente. De esta manera, hasta el momento hemos trabajado la noción de máquina social enfocándonos solamente en la propia máquina, reduciendo nuestra atención al área comprendida al interior de los límites de la máquina y los límites mismos. En esta sección terminamos de modelar el concepto de máquina social ampliando nuestra visión al cuadro general donde una máquina social participa de un entorno compuesto por distintas personas y posiblemente otras máquinas sociales, y donde todos pueden interactuar entre sí. El modelo que construiremos nos permitirá distinguir quiénes son los actores que interactúan en un determinado momento, cómo se sucede cada interacción y cómo estas interacciones afectan las posibles interacciones del resto de los actores. Una vez que integremos estas nociones al modelo, estaremos en condiciones también de expresar formalmente algunas de las propiedades más importantes y distintivas de las máquinas sociales. Para poder lograr todo esto, basaremos esta última sección del modelo en el π-cálculo, formalismo que se explica en la Sección 4.4 y que recomendamos al lector revisar en caso de considerarlo necesario. El π-cálculo nos proveerá de todas

las herramientas necesarias para modelar la comunicación entre los distintos actores. En efecto, cada *actor* tendrá asociado un proceso del π-cálculo que dará cuenta de su actividad. Las personas, que en adelante denominaremos *agentes*, se comunicarán libremente entre sí, por lo que podrán tener asociado cualquier proceso del π-cálculo. Por ejemplo, podemos asociar a un agente la expresión $A := \bar{c}z.c(y)$ para representar que éste espera enviar el nombre z por el canal c para luego esperar recibir un nombre por el mismo canal. De la misma forma, podemos asociar la expresión $B := c(x).\bar{c}x$ a otro agente para indicar que espera recibir un cierto nombre por el canal c para luego devolverlo por el mismo canal. De esta forma asociaremos a cada agente un cierto proceso del π-cálculo para describir su comportamiento. Las máquinas sociales, en cambio, tendrán otro trato. Primero que todo, seguirán una pauta estándar de comportamiento que, como veremos más adelante, no será más que la especificación de la actividad que hemos descrito en reiteradas oportunidades y que consiste en esperar recibir un input por el canal de entrada para luego disponer los outputs en los canales de salida. De esta forma los procesos asociados a las máquinas seguirán siempre el mismo patrón. En segundo lugar, para que un actor ya sea agente o máquina pueda comunicarse con una máquina, deberá primero *establecer una conexión* con ésta. Esto significa que el actor debe reservar un canal de comunicación disponible de la máquina social, el que en el futuro sólo podrá ser usado por ese actor y la máquina. En el modelo presentado en este trabajo, estas conexiones son de carácter *estático*, vale decir, la definición de los actores que participan del escenario que se estudia especifica también quién puede usar qué canal de cada máquina. En general, dada una máquina social $M_i = (Q_i, \Gamma_i, \Lambda_i, \delta_i, s_i, H_i)$, la máquina tiene todos los nombres de canales de salida en Λ_i a su disposición para comunicarse. Cuando un cierto agente establece una conexión con M reservando el canal λ_0, entonces ningún agente distinto de éste puede usar ese símbolo como nombre de canal. En el modelo incorporaremos esto a la inversa, vale decir, todos los agentes compartirán un conjunto de nombres base, a los que se unirán de forma particular los nombres de canales que cada uno reservó. Ahora bien, la comunicación con máquinas sociales seguirá un *protocolo de comunicación* específico. Establecida una conexión entre un actor y una cierta máquina M al reservar un nombre λ, para enviar un mensaje a M el proceso A asociado al actor deberá adoptar la siguiente forma:

$$A := \overline{\lambda}\langle w_i \rangle.P \qquad \text{(protocolo de envío de input)}$$

Donde w_i representa el input a enviar. A su vez, para recibir el output generado por M, el proceso asociado al actor deberá adoptar la siguiente forma:

$$A := \lambda(x).P \qquad \text{(protocolo de recibo de output)}$$

Al conjunto de los actores que son objeto de estudio, más las conexiones establecidas con las máquinas sociales, lo denominaremos *unidad de interacción social*. Usaremos esta noción como base para seguir desarrollando el modelo, debido a que define la *estructura* del entorno que se desea modelar. Luego de introducir formalmente esta noción nos preocuparemos de definir la *dinámica* de la unidad de interacción social, vale decir, cómo los distintos actores interactúan entre sí. Para hacer esto, pensaremos la dinámica como si se tratase de una obra de teatro. De antemano, tenemos la unidad de interacción social, que define a los actores y sus relaciones estructurales. Luego, asociaremos a cada actor un proceso que cumpla con las restricciones estructurales, lo que puede pensarse como el libreto de la obra. Llamaremos a esto la *pauta de interacción social*. Después, introduciremos la noción de *estado de interacción social*, que podemos verlo como una escena de la obra. El paso de una escena a

otra se corresponderá con una interacción entre los actores. La pauta de interacción social determinará el estado inicial de interacción social y en consecuencia el resto de los estados de interacción social; es decir, determina las secuencias de interacciones posibles entre los actores. Ahora bien, en estricto rigor, dada una cierta escena, el libreto de esta obra no especifica completamente la escena siguiente; esto debido a que de un mismo estado puede producirse más de una interacción posible, o en otras palabras, un mismo proceso en el π-cálculo tiene más de una reducción posible. Sin embargo, de todas formas esta metáfora da una buena intuición de cómo guiaremos el desarrollo de la sección. La Figura 5.7 ilustra un ejemplo de unidad de interacción social.

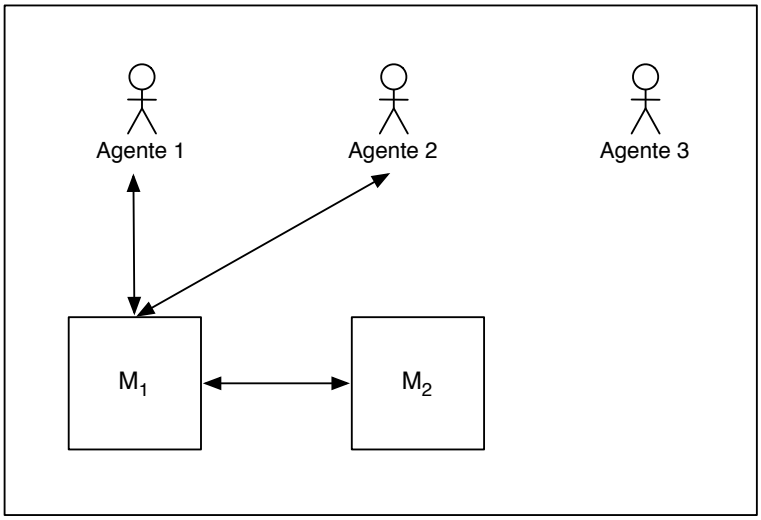


Figura 5.7: Unidad de interacción social.

En esta figura se ilustra un ejemplo que consta de 5 actores: 3 agentes y 2 máquinas. La flecha entre el Agente 1 y M_1 representa que este agente ha reservado una conexión con la máquina. Lo mismo ocurre para el Agente 2, quien también ha reservado una conexión con M_1. Nótese también la flecha entre M_1 y M_2, que representa que M_1 y M_2 han reservado una conexión entre sí. Esto significa que ambas máquinas usan un mismo nombre de canal para enviar y recibir mensajes entre sí, lo que implica que comparten el nombre de uno de sus canales de comunicación. Por último, la interacción entre los agentes se da libremente, es decir, no existe una restricción estructural que especifique el nombre de los posibles canales que usen para comunicarse entre sí.

Definición 5.38 *Una* Unidad de Interacción Social *es una tupla* $\mathcal{U} = (\Gamma, \mathcal{A}, \mathcal{M}, \mathcal{C})$ *donde*

- Γ *es un alfabeto finito tal que* $\sqcup, \# \in \Gamma$ *y* $\{L, R, \triangleright, \square\} \cap \Gamma = \emptyset$*;*
- $\mathcal{A}$ *es un conjunto finito no vacío de* agentes *(que representarán a las personas);*
- $\mathcal{M}$ *es un conjunto finito no vacío de máquinas sociales con índices en* I*;*

- $\mathcal{C} : \Lambda \nrightarrow \mathcal{A} \cup \mathcal{M}$ *es la* función parcial de conexión *donde* $\Lambda = \bigcup_{i \in I} \Lambda_i$*;*

De forma que se satisfacen las siguiente condiciones:

1. *Para todo* $i \in I$, $M_i = (Q_i, \Gamma_i, \Lambda_i, s_i, H_i)$ *es tal que* $\Gamma_i = \Gamma$ *y* $\Lambda_i \cap \Gamma = \emptyset$*;*

2. *Para todo* $i, j \in I$, $\Lambda_i \cap \Lambda_j = C^{-1}(M_i) \cap \Lambda_j \cup C^{-1}(M_j) \cap \Lambda_i$*;*

Donde $\mathcal{C}^{-1}(x) = \{\lambda \in \Lambda : \mathcal{C}(\lambda) = x\}$ *es el conjunto de preimágenes de* x *por* $\mathcal{C}$*. Finalmente, dada* $\mathcal{U}$ *una unidad de interacción social definimos* $\Sigma = \Gamma \cup \Lambda$*.*

Revisemos brevemente la definición anterior. El primer elemento que compone la unidad de interacción social corresponde al alfabeto Γ. Este alfabeto es el alfabeto base de comunicación de todas las máquinas sociales, y más adelante veremos que también es el alfabeto básico que comparten todos los agentes —excluyendo el blanco—. Es decir, representa intuitivamente, y con las variaciones correspondientes, el alfabeto común compartido por todos los actores de la unidad de interacción social. Más adelante veremos la forma exacta en que esto ocurre. El segundo componente de la definición lo conforman los agentes, que como ya habíamos anticipado representan a las personas que interactúan entre sí y con las máquinas sociales. Luego tenemos las máquinas sociales mismas, cada una indexada por un elemento $i \in I$. Después tenemos la función parcial de conexión, que asocia un identificador de canal de salida con un agente o una máquina. Esta función parcial se define sobre el conjunto de todos los nombres de canales de salida de todas las máquinas sociales en $\mathcal{M}$. Si un identificador λ es tal que $\lambda \in Dom(\mathcal{C})$, entonces diremos que λ está reservado por $\mathcal{C}(\lambda)$. Finalmente la condición *1* indica que todas las máquinas comparten el mismo alfabeto base de comunicación, a saber, Γ, mientras que ningún nombre de canal de salida es a la vez un símbolo en Γ; por su parte, la condición *2* indica que los únicos nombres de canal de salida que pueden compartir dos máquinas son los que fueron reservados por alguna de las dos. En particular, si una máquina social M_i no ha establecido conexiones con ninguna otra máquina social, entonces los nombres de sus canales de salida Λ_i le pertenecen solo a M_i.

Ejemplo 5.39 A continuación se ejemplifica la noción anterior describiendo una posible unidad de interacción social para la Figura 5.7. Sean M_1 y M_2 dos máquinas sociales tales que $\Lambda_1 = \{\lambda_1, \lambda_2, \lambda\}$ y $\Lambda_2 = \{\lambda\}$. Entonces una posible unidad de interacción social que ilustra la Figura 5.7 es $\mathcal{U} = (\Gamma, \mathcal{A}, \mathcal{M}, \mathcal{C})$ tal que:

$$\begin{aligned} \Gamma &= \{a, \ldots, z, \sqcup, \#\} \\ \mathcal{A} &= \{A_1, A_2, A_3\} \\ \mathcal{M} &= \{M_1, M_2\} \\ \mathcal{C} &= \{(\lambda_1, A_1), (\lambda_2, A_2), (\lambda, M_2)\} \end{aligned}$$

Donde $a, \ldots, z$ son las letras del alfabeto español. Así, en este caso tenemos que $\Lambda = \lambda_1, \lambda_2, \lambda$ y $\Sigma = \{a, \ldots, z, \lambda_1, \lambda_2, \lambda, \sqcup, \#\}$.

Ejemplo 5.40 En este ejemplo vamos a modelar la unidad de interacción social de una sala de chat. Más adelante, cuando formalicemos la dinámica de una unidad de interacción social retomaremos este ejemplo y describiremos posibles interacciones que pueden darse en la sala de chat. El funcionamiento de la sala de chat es muy sencillo: cada vez que un agente conectado a la sala de chat envía un mensaje, el resto de los agentes recibe este mensaje y la máquina social que provee el servicio lo almacena

anexándolo al historial de mensajes anteriores. Comenzamos primero definiendo la máquina social M_C que proveerá el servicio de la sala de chat. Para ello, defínanse los siguientes alfabetos:

$$\Gamma_C = \{0, \ldots, 9, a, \ldots, v, \sqcup, \#\}$$
$$\Lambda_C = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$$

Donde $0, \ldots, 9$ son los dígitos del sistema decimal y $a, \ldots, v$ son las letras de alfabeto español desde la a hasta la v. Luego, sea $\Sigma_0 = \Lambda_C \cup \Gamma_C - \{\#\}$ y defínase $f : \Sigma_0^* \times \Sigma_0^* \nrightarrow \Sigma_0^* \times (\Sigma_0^*)^4$ como:

$$f(w_i, w) = \begin{cases} (w \sqcup w_i, w_i, w_i, , w_i, w_i) & \text{si } w_i[1] \in \{\lambda_1, \lambda_2, \lambda_3, \lambda_4\} \\ \text{no definido} & \text{si no} \end{cases}$$

Entonces definimos M_C como la máquina social que p-computa f, tal que $\Gamma_{M_C} = \Gamma_C$ y $\Lambda_{M_C} = \Lambda_C$, y donde se ha dispuesto el siguiente orden de nombres de canales de salida: $\lambda_1, \lambda_2, \lambda_3, \lambda_4$. A continuación definimos la unidad de interacción social que modelará la sala de chat y sus actores. Esta unidad estará compuesta de tres agentes y la máquina M_C, de forma que cada agente estará conectado a la máquina. De este modo, definimos $\mathcal{U}_C$ como $\mathcal{U}_C = \{\Gamma, \mathcal{A}, \mathcal{M}, \mathcal{C}\}$ donde:

$$\Gamma = \{0, \ldots, 9, a, \ldots, v, \sqcup, \#\}$$
$$\mathcal{A} = \{A_1, A_2, A_3\}$$
$$\mathcal{M} = \{M_C\}$$
$$\mathcal{C} = \{(\lambda_1, A_1), (\lambda_2, A_2), (\lambda_3, A_3), \}$$

De esta forma tenemos que:

$$\Lambda = \Lambda_{M_C}$$
$$\Sigma = \Gamma \cup \Lambda = \Gamma_{M_C} \cup \Lambda_{M_C} = \Sigma_{M_C}$$

Nótese que la máquina tiene cuatro canales de comunicación a su disposición, de los cuales solo tres están reservados. Más adelante, cuando hayamos formalizado la dinámica de una unidad de interacción social, veremos algunos ejemplos de interacciones posibles que pueden llevarse a cabo en esta sala de chat.

A continuación introducimos las nociones que nos permitirán formalizar la interacción entre actores que participan en una determinada unidad de interacción social. Haremos esto usando como base el formalismo del π-cálculo, que nos entrega las herramientas necesarias para describir comunicación entre procesos. Como señalamos anteriormente, cada actor de la unidad de interacción social tendrá asociado un proceso del π-cálculo que describe su actividad. Sin embargo, tempranamente nos veremos enfrentados al hecho de que contar con estos procesos no es suficientes para modelar la dinámica de una unidad de interacción social, ya que necesitamos una forma de identificar los actores asociados a los nuevos procesos que irán surgiendo como producto de las reducciones. Por ejemplo, supongamos que tenemos tres agentes con procesos asociados:

$$a_1 := u(x) \qquad a_3 := u(x) \qquad a_3 := \bar{u}z$$

Supongamos también que queremos modelar la dinámica de esta unidad a través de la formula $a_1|a_2|a_3$. En principio parece una buena forma de modelar las posibles interacciones entre los actores; cada actor se representa por un proceso que corre en paralelo al resto, y la comunicación entre los actores se representa en términos de la relación de reducción. En

efecto, esta será la idea que nos guiará en nuestra formalización de la dinámica de la unidad de interacción social. Sin embargo, no podemos aplicar esta idea directamente sobre los procesos del π-cálculo. Consideremos la siguiente reducción:

$$a_1|a_2|a_3 \rightarrow u(x)$$

¿Entre qué actores ocurrió la interacción? ¿Cuál es el actor representado por $u(x)$? Nótese que la expresión de la derecha satisface $u(x) \equiv 0|u(x) \equiv 0|u(x)|0 \equiv 0|0|u(x)$. Debido a la regla STRUCT usada para definir la relación de reducción en el π-cálculo, cualquiera de estas expresiones equivalentes entre sí puede ser reemplazada libremente por $u(x)$ en la fórmula de la reducción. Es decir, para contestar estas preguntas, no podemos basarnos en el orden sintáctico de la expresión resultante, ni siquiera tenemos cómo exigir que se conserve el número de operandos de la composición paralela. Para resolver este problema introducimos en nuestro modelo una *gramática de actores* que nos permitirá describir formalmente las interacciones entre ellos aprovechando todo el potencial del π-cálculo y sin perder la identidad de los procesos asociados luego de que se reducen. Definimos esta gramática a continuación.

Definición 5.41 *Sea $\mathcal{U} = (\Gamma, \mathcal{A}, \mathcal{M}, \mathcal{C})$ una unidad de interacción social y $\mathcal{P} = \mathcal{P}(\mathcal{N})$[4] con $\mathcal{N} = (\Sigma - \{\#\})^*$. Se define el conjunto $\mathcal{E}$ de expresiones de actores por medio de la siguiente gramática:*

$$E ::= \ (x, P) \ \Big| \ E_1 \mid E_2$$

Donde $x \in \mathcal{A} \cup \mathcal{M}$ y $P \in \mathcal{P}$. Usaremos la notación $\mathcal{E}(\mathcal{U})$ para referirnos al conjunto de expresiones de actores definido sobre la unidad de interacción social $\mathcal{U}$.

Como podemos ver, la gramática de actores es muy sencilla. Su primitiva es una expresión de la forma (x, P) donde x es un agente o una máquina y P es un proceso. Esta expresión obliga a escribir en cada momento la identidad del actor seguido del proceso que indica su actividad presente. En la Figura 5.8 se ilustra este punto.

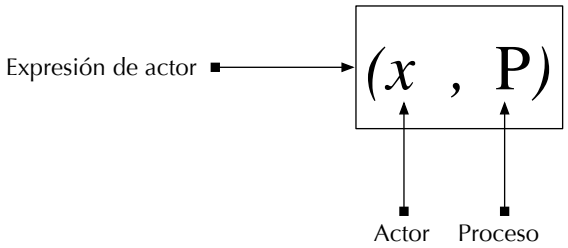


Figura 5.8: Expresión de actor primitiva.

Luego tenemos la operación de *composición paralela*, que es una adaptación de la composición paralela del π-cálculo a la gramática de actores. Así la expresión $(x, P)|(y, Q)$ representa dos actores de identidad x e y cuya actividad respectiva P y Q se desarrolla simultáneamente, y en general la expresión $E_1|E_2$ representa el grupo de actores en E_1 y E_2 desarrollando su actividad de forma simultánea.

[4]Recuérdese que en la Definición 4.17 introdujimos la notación $\mathcal{P}(\mathcal{N})$ para denotar el conjunto de expresiones de procesos del π-cálculo con conjunto de nombres $\mathcal{N}$.

Al igual que sucede con el π-cálculo, expresiones sintácticamente distintas pueden representar en realidad el mismo concepto. Este sentido de equivalencia entre expresiones de actores lo formalizamos a continuación definiendo en $\mathcal{E}$ una relación de congruencia estructural. Recuérdese que usamos el símbolo $\equiv$ para denotar la congruencia estructural del π-cálculo.

Definición 5.42 *Se define la relación de* congruencia estructural *en $\mathcal{E}$, símbolo $\cong$, como la congruencia más pequeña sobre $\mathcal{E}$ que satisface las siguientes ecuaciones:*

1. $(x, P) \cong (x, Q)$ *cada vez que* $P \equiv Q$;
2. $E_1 \mid E_2 \cong E_2 \mid E_1$;
3. $(E_1 \mid E_2) \mid E_3 \cong E_1 \mid (E_2 \mid E_3)$.

En la siguiente definición extendemos la relación de reducción del π-cálculo a la gramática de actores que hemos introducido. De esta forma podremos escribir las reducciones de los procesos asociados a cada actor, pero ahora en términos de reducciones sobre expresiones de actores. Esto nos permitirá contar con todas las herramientas que entrega el π-cálculo a la vez que seremos capaces de reconocer a quién representa un proceso incluso después de reducciones. Presentamos esta definición a continuación.

Definición 5.43 *Se define la relación de* reducción *en $\mathcal{E}$, símbolo $\rightarrow_{\mathcal{E}}$, como la relación más pequeña que satisface las siguientes reglas:*

SING: $\quad (x, P) \rightarrow_{\mathcal{E}} (x, Q) \quad$ *si y solo si* $P \rightarrow Q$

COM: $\quad (a, u(y).P) \mid (b, \bar{u}z.Q) \rightarrow_{\mathcal{E}} (a, P\{z/y\}) \mid (b, Q)$

PAR:
$$\frac{E \rightarrow_{\mathcal{E}} E'}{E \mid D \rightarrow_{\mathcal{E}} E' \mid D}$$

STRUCT:
$$\frac{D \equiv E \quad E \rightarrow_{\mathcal{E}} E' \quad E' \equiv D'}{D \rightarrow_{\mathcal{E}} D'}$$

Denotaremos por $\rightarrow_{\mathcal{E}}^*$ la clausura refleja y transitiva de $\rightarrow_{\mathcal{E}}$.

Revisemos brevemente esta definición. La primera regla, SING, considera el caso de la actividad interna de un actor. Si el proceso asociado a un actor es reductible, entonces la expresión de actor que asocia a ese actor ese proceso debe también ser reductible. La segunda regla, COM, considera el caso de interacción entre actores distintos. Esta es la regla que captura la esencia de lo que hemos estado buscando: dos actores distintos con sus respectivos procesos asociados, procesos que luego de la reducción siguen estando asociados a los actores originales. Finalmente, la tercera regla PAR y la cuarta STRUCT son análogas a las del π-cálculo y responden al significado mismo de la composición paralela y de la congruencia estructural, respectivamente.

Ahora que contamos con los elementos necesarios para modelar las interacciones entre

los distintos actores, corresponde definir la pauta de interacción estándar que seguirán las máquinas sociales, es decir, el proceso del π-cálculo que representará el comportamiento que cada máquina exhibe en su entorno.

Definición 5.44 *Sea $\mathcal{U} = (\Gamma, \mathcal{A}, \mathcal{M}, \mathcal{C})$ una unidad de interacción social, $M \in \mathcal{M}$ una máquina social definida como $M = (Q, \Gamma, \Lambda_M, \delta, s, H)$ y $\mathcal{P} = \mathcal{P}(\mathcal{N})$. Se define el* proceso estándar de M en el π-cálculo*, notación M_π, como el proceso $M_\pi \in \mathcal{P}$ definido de la siguiente forma (la notación ha sido ligeramente modificada para facilitar la lectura):*

$$M_\pi\langle w\rangle := \ (a).\sum_{\lambda\in\overline{\Lambda}_M} \lambda(x).[\ \bar{a}\langle \mathrm{cat}_{(\lambda,x)}\rangle \mid a(z).(\ \mathrm{out}^M_{(z,w)} \mid M_\pi\langle \mathrm{state}^M_{(z,w)}\rangle\ )\ ]$$

Donde $\overline{\Lambda}_M$ se define como $\overline{\Lambda}_M = \Lambda_M \cap Dom(\mathcal{C})$ y representa el conjunto de los canales reservados de M. También, la función cat *se define como:*

$$\mathrm{cat}_{(x,y)} = x \sqcup y$$

y las funciones out *y* state *se definen como sigue. Sea $w \vDash_M w'$ $[w_i, w_o]$ una computación persistente de M, entonces:*

$$\mathrm{out}^M_{(w_i,w)} := \prod_{\lambda\in\overline{\Lambda}_M} \bar{\lambda}\langle w_o(\lambda)\rangle$$
$$\mathrm{state}^M_{(w_i,w)} = w'$$

También nos referiremos al proceso estándar de M en el π-cálculo como la π-máquina de M. Algunas veces no haremos distinción entre el proceso estándar de M y la máquina social M, si ello no lleva a confusiones.

Revisemos brevemente esta definición. Primero vemos que el proceso estándar de una máquina social se define paramétricamente en función del nombre w. Este nombre representará el string de estado inicial de la máquina, y será utilizado para calcular los valores de output y estado final una vez que el proceso haya recibido el input. El proceso reserva el nombre a para su uso exclusivo, aunque este nombre no cumple otra función más que solamente simplificar la expresión del proceso estándar. La π-máquina comienza recibiendo de forma no determinista un mensaje a través de algunos de los canales que tiene reservados de acuerdo a la función parcial de conexión. Este no determinismo, denotado por la sumatoria, expresa con fidelidad el hecho de que la máquina desconoce qué actor intentará comunicarse primero con ella, pero una vez que la máquina recibe un input, no vuelve a esperar otro input del entorno hasta finalizar la p-computación. Una vez que recibe el input, la máquina concatena el identificador del canal al input recibido, generando un string de la forma $\lambda \sqcup w_i$. Asumiremos este tipo de entrada para todas las máquinas sociales que se representen a través de su proceso estándar. Esto permitirá que la máquina siempre pueda acceder a la información sobre quién es el emisor del mensaje, y por tanto discriminar entre los distintos actores del entorno que interactúan con ella. Finalmente el proceso termina con dos subprocesos en paralelo. Uno espera enviar los outputs por los canales de salida, mientras que el otro corresponde a la π-máquina definida esta vez con parámetro igual al string de estado final de la p-computación que indujo el input que recibió del entorno y el parámetro w inicial. De esta forma, vemos cómo esta expresión permite modelar el comportamiento externo de las máquinas sociales, incorporando de forma coherente las partes del modelo que desarrollamos en las secciones anteriores.

Ejemplo 5.45 Veamos a través de un ejemplo cómo se comporta el proceso estándar de una máquina social. Tomemos como ejemplo la máquina social M_{Latch} introducida en el Ejemplo 5.12 y consideremos la p-computación $\varepsilon \vDash 1\ [1101, 0]$. Por lo mencionado en el párrafo anterior, debemos modificar ligeramente esta p-computación para considerar en el input el identificador reservado por el emisor; para ello asumamos que la p-computación es $\varepsilon \vDash 1\ [\lambda \sqcup 1101, 0]$. Ahora bien, supongamos que solo existen dos actores, un agente a y la propia M_{Latch}, de forma que la expresión de actores que describe el estado de la interacción es:

$$(a, \bar{\lambda}\langle 1101\rangle.\lambda(x)) \mid (M, M_\pi\langle\varepsilon\rangle)$$

Donde omitimos el subíndice de M_{Latch}. Entonces esta expresión se puede reducir como sigue:

$$\begin{aligned}
& (a, \bar{\lambda}\langle 1101\rangle.\lambda(x)) \mid (M, M_\pi\langle\varepsilon\rangle) \\
= {} & (a, \bar{\lambda}\langle 1101\rangle.\lambda(x)) \mid (M, (a).\lambda(x).[\ \bar{a}\langle \text{cat}_{(\lambda,x)}\rangle \mid a(z).(\ \text{out}^M_{(z,w)} \mid M_\pi\langle \text{state}^M_{(z,w)}\rangle\ )\ ]) \\
\to_{\mathcal{E}} {} & (a, \lambda(x)) \mid (M, (a).[\ \bar{a}\langle \text{cat}_{(\lambda,1101)}\rangle \mid a(z).(\ \text{out}^M_{(z,\varepsilon)} \mid M_\pi\langle \text{state}^M_{(z,\varepsilon)}\rangle\ )\ ]) \\
\to_{\mathcal{E}} {} & (a, \lambda(x)) \mid (M, (a).[\ \text{out}^M_{(\lambda\sqcup 1101,\varepsilon)} \mid M_\pi\langle \text{state}^M_{(\lambda\sqcup 1101,\varepsilon)}\rangle\ ]) \\
= {} & (a, \lambda(x)) \mid (M, (a).[\ \bar{\lambda}0 \mid M_\pi\langle 1\rangle\ ]) \\
\to_{\mathcal{E}} {} & (a, 0) \mid (M, (a).M_\pi\langle 1\rangle\ ) \\
\cong {} & (a, 0) \mid (M, M_\pi\langle 1\rangle\ )
\end{aligned}$$

De esta forma hemos probado que:

$$(a, \bar{\lambda}\langle 1101\rangle.\lambda(x)) \mid (M, M_\pi\langle\varepsilon\rangle) \quad \to^*_{\mathcal{E}} \quad (a, 0) \mid (M, M_\pi\langle 1\rangle)$$

Y en el proceso M ha respondido ante el input $\lambda \sqcup 1101$ y con estado inicial ε, con el output 0 y quedando con estado final 1. Es decir, ha llevado a cabo la p-computación $\varepsilon \vDash 1\ [\lambda \sqcup 1101, 0]$.

En lo que sigue concluimos la formalización de la dinámica de la unidad de interacción social para pasar luego a explorar la descripción de algunas de las propiedades más características de las máquinas sociales. En específico, respecto al primer punto, nos quedan dos tareas por cumplir. La primera, corresponde a la definición de la pauta de interacción social, que define el *libreto* bajo el cual se llevan a cabo las distintas interacciones. Esta pauta asocia a cada actor un determinado proceso que debe cumplir con las restricciones estructurales propias de la unidad de interacción social de la que participa. La segunda tarea corresponde a la definición de los estados de interacción social, que nos permitirán dar cuenta de los escenarios posibles que pueden resultar de la interacción entre los distintos actores. Presentamos a continuación estas definiciones.[5]

Definición 5.46 *Sea $\mathcal{U} = (\Gamma, \mathcal{A}, \mathcal{M}, \mathcal{C})$ una unidad de interacción social y $\Sigma = \Gamma \cup \Lambda$. Definimos una* pauta de interacción social *de $\mathcal{U}$ como una función $g : (\mathcal{A} \cup \mathcal{M}) \to \mathcal{P}$ tal que se satisface:*

1. *$\mathcal{P} = \mathcal{P}(\mathcal{N})$ con conjunto de nombres $\mathcal{N} = (\Sigma - \{\#\})^*$;*

2. *Para todo $a \in \mathcal{A}$ se tiene que $\mathcal{K}(g(a)) \subset (\Gamma - \{\#\})^* \cup \mathcal{C}^{-1}(a)$;*

[5]Se recuerda al lector que la notación $\mathcal{K}(P) \subset \mathcal{N}_0$ fue introducida en la Definición 4.20 para denotar que el proceso P tiene sus canales de interacción en el conjunto de nombres $\mathcal{N}_0$.

3. *Para todo* $M \in \mathcal{M}$ *se tiene que* $g(M) := M_\pi\langle\varepsilon\rangle$*;*

4. *Para todo* $M \in \mathcal{M}$ *se tiene que* $\mathcal{K}(g(M)) \subset \overline{\Lambda}_M$*.*

Observación 5.47 Nótese que el punto *4* de la Definición 5.46 no aporta mayor información debido a la propia definición de proceso estándar de una máquina en el π-cálculo. Sin embargo la inclusión de este punto resulta fundamental si en el futuro se quisieran modelar algunas máquinas sociales a través de un comportamiento distinto al estándar.

Definición 5.48 *Sea* $\mathcal{U} = (\Gamma, \mathcal{A}, \mathcal{M}, \mathcal{C})$ *una unidad de interacción social,* $\mathcal{E} = \mathcal{E}(\mathcal{U})$ *y* g *una pauta de interacción social de* $\mathcal{U}$*. Se define el* estado inicial de interacción social *de* $\mathcal{U}$ *dado* g*,* $E_g \in \mathcal{E}$ *como:*

$$E_g := \prod_{a\in\mathcal{A}} (a, g(a)) | \prod_{M\in\mathcal{M}} (M, g(M))$$

De forma general, definimos un estado de interaccion social como cualquier expresión de actores $E \in \mathcal{E}$ *tal que:*

$$E_g \to_{\mathcal{E}}^* E$$

Por último, dada una secuencia de reducciones de la forma $s = E_1 \to_{\mathcal{E}} E_2, \ldots, E_{n-1} \to_{\mathcal{E}} E_n$ *decimos que* s *está en* $\mathcal{U}$ *si y solo si* $E_g \to_{\mathcal{E}}^* E_1$*.*

Ejemplo 5.49 A continuación retomamos el Ejemplo 5.40 de la sala de chat y modelamos algunas de las interacciones que pueden llevarse a cabo en ella. Estábamos considerando la unidad de interacción social $\mathcal{U}_C$ definida como $\mathcal{U}_C = \{\Gamma, \mathcal{A}, \mathcal{M}, \mathcal{C}\}$ donde:

$$\begin{aligned}\Gamma &= \{0, \ldots, 9, a, \ldots, v, \sqcup, \#\}\\ \mathcal{A} &= \{A_1, A_2, A_3\}\\ \mathcal{M} &= \{M_C\}\\ \mathcal{C} &= \{(\lambda_1, A_1), (\lambda_2, A_2), (\lambda_3, A_3), \}\end{aligned}$$

Y donde M_C es la máquina social que implementa el servicio con $\Lambda_{M_C} = \{\lambda_1, \lambda_2, \lambda_3, \Lambda_4\}$. Dicho esto, ahora definimos la pauta de interacción social g de $\mathcal{U}_C$ como $g : (\mathcal{A} \cup \mathcal{M}) \to \mathcal{P}$ tal que:

$$\begin{aligned}g(M_C) &:= M_{C\pi}\langle\varepsilon\rangle\\ g(A_1) &:= \bar{\lambda}_1\langle hola\rangle.\lambda_1(x).\overline{2266637}\langle lee \sqcup el \sqcup chat\rangle.P_1\\ g(A_2) &:= \lambda_2(x).\bar{\lambda_2}\langle hola\rangle.P_2\\ g(A_3) &:= 2266637(x).P_3 \ + \ 92305160(y).\overline{92305160}\langle alo\rangle.Q_3\end{aligned}$$

Podemos interpretar esta pauta de interacción como sigue. M_C seguirá la pauta estándar de actividad de las máquinas sociales. El agente A_1 espera enviar «hola» a la sala de chat, luego esperará recibir un mensaje de la sala de chat, luego enviará el mensaje «lee el chat» por el canal de nombre 2266637 —que ha sido nombrado así para sugerir un llamado telefónico y donde interpretamos el símbolo $\sqcup$ como un espacio— y luego seguirá según P_1. El agente A_2 espera recibir un mensaje de la sala de chat, luego esperará a enviar un «hola» a la sala de chat, y luego seguirá según P_2. Por último, el agente A_3 espera que le envíen un mensaje o bien por el canal 2266637 o bien por el canal 92305160. Si recibe el mensaje por el canal 2266637, entonces seguirá según P_3, mientras que si recibe un mensaje por el otro canal, contestará con un «alo» y seguirá según Q_3 —podemos interpretar esta

última actividad como una persona que espera recibir un llamado ya sea a su teléfono fijo o a su teléfono móvil, y que dependiendo del canal por el que escuche seguirá una u otra actividad—.

Dada esta pauta g de interacción social se tiene que el estado inicial de interacción social E_g queda definido como:

$$E_g = (A_1, g(A_1)) \mid (A_2, g(A_2)) \mid (A_3, g(A_3)) \mid (M_C, M_{C\pi}\langle\varepsilon\rangle)$$

A continuación, damos algunos ejemplos de interacciones que pueden darse en la sala de chat. Por ejemplo, una interacción entre A_1 y M_C queda representada por la secuencia de reducciones de la forma:

$$\begin{aligned} E_g \to_{\mathcal{E}}^{*} E_1 = &(A_1, \lambda_1(x).\overline{2266637}\langle lee \sqcup el \sqcup chat\rangle.P_1) \mid \\ &(A_2, g(A_2)) \mid \\ &(A_3, g(A_3)) \mid \\ &(M_C, \prod_{1,2,3} \bar{\lambda}_i\langle\lambda_1 \sqcup hola\rangle | M_{C\pi}\langle\lambda_1 \sqcup hola\rangle) \end{aligned}$$

Como vemos, la máquina social M_C ha realizado una p-computación disponiendo en cada canal de salida el output correspondiente, y almacenando como estado el historial actualizado de mensajes de la sala de chat —que hasta ahora corresponde al único mensaje enviado—. El agente A_1, por su parte, ahora espera recibir un mensaje de la sala de chat. Las interacciones entre los actores de la unidad de interacción social podría continuar llevándolos, por ejemplo, al siguiente estado E_2:

$$\begin{aligned} E_1 \to_{\mathcal{E}}^{*} E_2 = &(A_1, \overline{2266637}\langle lee \sqcup el \sqcup chat\rangle.P_1) \mid \\ &(A_2, \bar{\lambda}_2\langle hola\rangle.P_2) \mid \\ &(A_3, 2266637(x).P_3 \ + \ 92305160(y).\overline{92305160}\langle alo\rangle.Q_3) \mid \\ &(M_C, \bar{\lambda}_3\langle\lambda_1 \sqcup hola\rangle | M_{C\pi}\langle\lambda_1 \sqcup hola\rangle) \end{aligned}$$

La actividad de los actores descrita en esta secuencia de estados, indica que los agentes A_1 y A_2 recibieron un mensaje de la sala de chat. Por su parte M_C aún espera enviar un mensaje a A_3. Finalmente ejemplificamos el hecho de que un mismo estado puede ser testigo de más de una interacción posible, y por lo tanto puede reducirse a más de un estado de interacción social. Así, por ejemplo podría suceder que el agente P_1 se comunique con el agente P_3 enviándole el mensaje «lee el chat» por el canal 2266637, interacción modelada por la siguiente reducción:

$$\begin{aligned} E_2 \to_{\mathcal{E}} E_3 = &(A_1, P_1) \mid \\ &(A_2, \bar{\lambda}_2\langle hola\rangle.P_2) \mid \\ &(A_3, P_3) \mid \\ &(M_C, \bar{\lambda}_3\langle\lambda_1 \sqcup hola\rangle | M_{C\pi}\langle\lambda_1 \sqcup hola\rangle) \end{aligned}$$

O bien, podría suceder que el agente P_2 enviara el mensaje «hola» a la máquina social M_C, de forma que ésta realice una p-comutación, disponiendo los outputs generados en los canales de salida respectivos y almacenando como string de estado el historial actualizado de los mensajes de la sala de chat:

$$\begin{aligned} E_2 \to_{\mathcal{E}}^{*} E_4 = &(A_1, \overline{2266637}\langle lee \sqcup el \sqcup chat\rangle.P_1) \mid \\ &(A_2, P_2) \mid \\ &(A_3, 2266637(x).P_3 \ + \ 92305160(y).\overline{92305160}\langle alo\rangle.Q_3) \mid \\ &(M_C, \bar{\lambda}_3\langle\lambda_1 \sqcup hola\rangle | \prod_{1,2,3} \bar{\lambda}_i\langle\lambda_2 \sqcup hola\rangle | M_{C\pi}\langle\lambda_1 \sqcup hola \sqcup \lambda_2 \sqcup hola\rangle) \end{aligned}$$

Nótese que en este último caso, en el estado de interacción social resultante E_4 la máquina social M_C espera enviar al agente A_3 tanto el nuevo output generado como el output anterior que éste aún no ha recibido.

En lo que resta del modelo nos interesaremos en poder expresar algunas de las propiedades más relevantes de las máquinas sociales. Para ello nos serán muy útiles las siguientes definiciones.

Definición 5.50 *Sea $\mathcal{U} = (\Gamma, \mathcal{A}, \mathcal{M}, \mathcal{C})$ una unidad de interacción social, $\mathcal{E} = \mathcal{E}(\mathcal{U})$ y g una pauta de interacción social de $\mathcal{U}$. Sea $a_0 \in \mathcal{A}$ un agente cualquiera y $M_0 \in \mathcal{M}$ una máquina social cualquiera. Sea C la expresión de actores siguiente:*

$$C := \prod_{\substack{a \in A \\ a \neq a_0}} (a, P_a) | \prod_{\substack{M \in \mathcal{M} \\ M \neq M_0}} (M, P_M)$$

Y considérense los siguientes estados de interacción social:

$$E_1 \cong C \mid (a_0, \bar{\lambda}_0\langle w_i \rangle.Q) \mid (M_0, M_{0\pi}\langle w \rangle)$$

$$E_2 \cong C \mid (a_0, Q) \mid (M_0, (a).[\ \bar{a}\langle \lambda_0 \sqcup w_i \rangle \mid a(z).(\ \mathrm{out}^{M_0}_{(z,w)} \mid M_{0\pi}\langle \mathrm{state}^{M_0}_{(z,w)} \rangle)])$$

$$E_3 \cong C \mid (a_0, Q) \mid (M_0, [\ \bar{w_o} \mid M_{0\pi}\langle w' \rangle\ ])$$

Donde $\lambda_0 \in \bar{\Lambda}_{M_0}$, $w' = \mathrm{state}^{M_0}_{(\lambda_0 \sqcup w_i, w)}$, y $\bar{w_o} = \mathrm{out}^{M_0}_{(\lambda_0 \sqcup w_i, w)}$. Entonces definimos una acción persona-máquina como una secuencia de reducciones de la forma:

$$E_1 \to_{\mathcal{E}} E_2\ ,\ E_2 \to_{\mathcal{E}} E_3$$

Lo que anotaremos de la siguiente manera:

$$a_0 \curvearrowright M_0 \qquad \lambda_0 :: w \vDash w'[w_i, w_o]$$

Y donde permitiremos escribir escribir la función w_o como una secuencia de strings, dado un ordenamiento previo de los identificadores de canales de salida. También, considérense los siguientes estados de interacción social:

$$D_1 \cong C \mid (a_0, \lambda_0(x).Q) \mid (M_0, [\prod_{\lambda \in \overline{\Lambda}_M} \bar{\lambda}\langle w_o(\lambda) \rangle | M_{0\pi}\langle w \rangle])$$

$$D_2 \cong C \mid (a_0, Q\{w_{\lambda_0}/x\}) \mid (M_0, [\prod_{\substack{\lambda \in \overline{\Lambda}_M \\ \lambda \neq \lambda_0}} \bar{\lambda}\langle w_o(\lambda) \rangle | M_{0\pi}\langle w \rangle])$$

Donde $\lambda_0 \in \bar{\Lambda}_{M_0}$ *y* $w_{\lambda_0} = w_o(\lambda_0)$. *Entonces definimos una acción máquina-persona como una reducción de la forma:*

$$D_1 \to_{\mathcal{E}} D_2$$

Lo que anotaremos a su vez de la siguiente manera:

$$M_0 \looparrowright a_0 \qquad \lambda_0 :: w_{\lambda_0}$$

Revisemos brevemente la definición anterior. Una acción persona-máquina corresponde a una secuencia de reducciones que representan una máquina social que recibiendo un input por parte de un agente lleva a cabo una p-computación. Una vez realizada la p-computación, la máquina dispone sus outputs en los canales de salida a la espera de que algún agente esté escuchando por un canal correspondiente. La expresión C de la definición anterior representa al resto de los actores de la unidad de interacción social, quienes no participan de la acción persona-máquina especificada. Por eso agrupamos todos los términos inactivos en una sola expresión, con lo que también se simplifican bastante las ecuaciones. Nótese que para modelar una p-computación a través del proceso estándar de una máquina se necesitan dos reducciones, por lo cual fue necesario describir tres estados de interacción social. Nótese también que hemos usado la noción de congruencia para asegurar que la definición de acción considere expresiones sintácticamente distintas pero que representan una misma situación. Luego introducimos la definición de acción máquina-persona. Esta acción corresponde sencillamente al recibo de un output de la máquina por parte de un agente. Puede modelarse usando una sola reducción, por lo que solo fue necesario especificar dos estados de interacción social. Finalmente, obsérvese que la notación escogida para denotar estos conceptos incluye la información sobre los canales usados en la interacción, añadiendo en el caso de la acción persona-máquina la p-computación misma, y en el caso de la acción máquina-persona el output enviado.

Observación 5.51 Considérese la siguiente secuencia de reducciones que han sido escritas verticalmente por razones de comodidad:

$$\begin{aligned} M &\looparrowright a \qquad \lambda_a :: w_a \\ M &\looparrowright b \qquad \lambda_b :: w_b \\ M &\looparrowright c \qquad \lambda_c :: w_c \\ M &\looparrowright d \qquad \lambda_d :: w_d \end{aligned}$$

En adelante permitiremos escribir secuencias como las anteriores usando la siguiente notación:

$$M \looparrowright a, b, c, d \qquad \lambda_a, \lambda_b, \lambda_c, \lambda_d :: w_a, w_b, w_c, w_d$$

También, si dada un acción persona-máquina queremos enfatizar la presencia de otros actores que participan del estado de interacción social, usaremos la siguiente notación:

$$a, b, c, d \looparrowright M_0 \qquad \lambda_a :: w \vDash w'[w_i, w_o]$$

Donde especificaremos con claridad el canal asociado al actor que realiza la acción persona-máquina.

Las propiedades de las máquinas sociales que consideramos a continuación forman parte del estudio que hicimos en la primera parte de esta memoria. Partimos expresando en nuestro

modelo la noción de consumo puro. Un acto de consumo puro de un servicio ofrecido por una máquina social podemos caracterizarlo por la ausencia total de contribución al mismo servicio. Esto significa que el servicio debe quedar inalterado luego del acto de consumo puro, es decir, la interacción entre el usuario y la máquina no deben modificar el comportamiento de esta última. Esto podemos expresarlo en nuestro modelo a través de una acción persona-máquina que no modifica el string de estado, es decir:

$$a \looparrowright M \qquad \lambda :: w \vDash w\ [w_i, w_o] \qquad \text{(consumo puro)}$$

Pese a que la acción del agente puede ser gatillante para que otros agentes distintos modifiquen la máquina, la p-computación vista atómicamente deja a la máquina con la misma estructura interna, y dispuesta a comportarse de la misma forma que si a no hubiera actuado sobre ella.

Por otro lado, un acto de producción pura implica necesariamente que la máquina social vea modificado su contenido. Más aún, la producción pura se caracteriza por la ausencia total de consumo, lo que significa que el servicio que ofrece la máquina debe omitirse durante esta interacción. De esta forma podemos expresar la noción de producción pura a través de la siguiente acción persona-máquina:

$$\begin{aligned} &a \looparrowright M \qquad \lambda :: w \vDash w'\ [w_i, w_o] \qquad \text{(producción pura)} \\ &w \neq w' \quad \wedge \quad Img(w_o) = \{\varepsilon\} \end{aligned}$$

Esto significa que la máquina termina con un estado distinto al inicial después de la p-computación, y que todo agente conectado a la máquina que intente sacar provecho de la contribución solo leerá el string vacío.

Finalmente, la noción de prosumo se expresa como una interacción que consume y produce a la vez, es decir cualquier acción distinta al consumo puro y la producción pura. Así el prosumo queda expresado como:

$$\begin{aligned} &a \looparrowright M \qquad \lambda :: w \vDash w'\ [w_i, w_o] \qquad \text{(prosumo)} \\ &w \neq w' \quad \wedge \quad Img(w_o) \neq \{\varepsilon\} \end{aligned}$$

Analicemos brevemente esta acción. Primero, esta acción consume el servicio de la máquina social debido a que se genera un output no vacío para algún agente. Sin embargo, no es una acción de consumo puro porque el string de estado de la máquina sí se modifica. Esto último a su vez implica que el agente modificó el contenido de la máquina, por lo que está produciendo, pero no es producción pura debido que alguien está recibiendo el fruto de la p-computación.

A continuación concentraremos nuestra atención en modelar tres clases distintas de contribución que los usuarios pueden mantener con una máquina social. Estas corresponden a la contribución directa, indirecta y obligada. En una contribución directa el usuario interactúa directamente con la máquina social que recibe la contribución. Este es el caso típico de contribución, y como ejemplo podemos mencionar a Wikipedia, World of Warcraft y Linux, entre otros. Luego tenemos las contribuciones del tipo indirecta. En este tipo de contribución los usuarios interactúan a través de un sistema o servicio que hace visible estas interacciones a un segundo sistema distinto y que las recibe en forma de contribuciones. Este es el caso por ejemplo de los GWAPs y las aplicaciones dependientes como el recomendador de libros

de Amazon. Finalmente tenemos las contribuciones obligadas que son aquellas que se exigen como forma de pago para poder acceder a otros servicios. El caso emblemático aquí es re-CAPTCHA. El primer tipo de contribución podemos expresarlo sin mayor complicación de la siguiente forma:

$$\begin{array}{ll} a \looparrowright M & \lambda :: w \vDash w' \ [w_i, w_o] \\ w \neq w' & \end{array} \qquad \text{(contribución directa)}$$

Nótese que estamos asumiendo una contribución como una interacción que altera el estado de la máquina, es decir, la contribución es efectiva, y por lo tanto es también un acto de producción. Para expresar el segundo tipo de contribuciones debemos considerar la interacción entre dos máquinas sociales. Para ello extenderemos las nociones de acción entre actores al caso de máquinas sociales.

Definición 5.52 *Sea $\mathcal{U} = (\Gamma, \mathcal{A}, \mathcal{M}, \mathcal{C})$ una unidad de interacción social, $\mathcal{E} = \mathcal{E}(\mathcal{U})$ y g una pauta de interacción social de $\mathcal{U}$. Sean $M_0, M_1 \in \mathcal{M}$ dos máquinas sociales cualquiera. Sea C la expresión de actores introducida en la Definición 5.50, y considérense los siguientes estados de interacción social:*

$$E_1 \cong C \mid (M_0, [\prod_{\lambda \in \overline{\Lambda}_M} \bar{\lambda}\langle w_o(\lambda)\rangle | M_{0\pi}\langle w\rangle]) \mid (M_1, M_{1\pi}\langle u\rangle)$$

$$E_2 \cong C \mid (M_0, [\prod_{\substack{\lambda \in \overline{\Lambda}_M \\ \lambda \neq \lambda_0}} \bar{\lambda}\langle w_o(\lambda)\rangle | M_{0\pi}\langle w\rangle]) \mid (M_1, (a).[\ \bar{a}\langle \lambda_0 \sqcup w_{\lambda_0}\rangle \mid a(z).(\ \mathrm{out}^{M_1}_{(z,u)} \mid M_{1\pi}\langle \mathrm{state}^{M_1}_{(z,u)}\rangle)])$$

$$E_3 \cong C \mid (M_0, [\prod_{\substack{\lambda \in \overline{\Lambda}_M \\ \lambda \neq \lambda_0}} \bar{\lambda}\langle w_o(\lambda)\rangle | M_{0\pi}\langle w\rangle]) \mid (M_1, [\ \bar{v_o} \mid M_{1\pi}\langle u'\rangle\ ]\ )$$

Donde $\lambda_0 \in \bar{\Lambda}_{M_1}$, $u' = \mathrm{state}^{M_1}_{(\lambda_0 \sqcup w_{\lambda_0}, u)}$, y $\bar{v_o} = \mathrm{out}^{M_1}_{(\lambda_0 \sqcup w_{\lambda_0}, u)}$. Entonces definimos una acción máquina-máquina de M_0 a M_1 como una secuencia de reducciones de la forma:

$$E_1 \to_{\mathcal{E}} E_2 \ , \ E_2 \to_{\mathcal{E}} E_3$$

Lo que anotaremos de la siguiente manera:

$$M_0 \looparrowright M_1 \qquad \lambda_0 :: u \vDash u' \ [w_{\lambda_0}, v_o]$$

Observación 5.53 Para incorporar la noción de contribución indirecta haremos uso de la definición de acción máquina-máquina, e introduciremos un máquina social en formato no-estándar. El formato que seguirá esta máquina será, sin embargo, muy similar al estándar; la única variación que haremos será limitar sus canales de salida. De esta forma, sea R el proceso del π-cálculo con que se define el proceso estándar de una máquina social en la Definición 5.44. Entonces denotaremos por $M\langle w, \lambda_0\rangle$ el proceso definido como $M\langle w, \lambda_0\rangle = R$, pero donde la función out se define como:

$$\mathrm{out}^{M}_{(w_i, w)} := \prod_{\substack{\lambda \in \overline{\Lambda}_M \\ \lambda \neq \lambda_0}} \bar{\lambda}\langle w_o(\lambda)\rangle$$

Es decir, la única diferencia de este proceso $M\langle w, \lambda_0\rangle$ con el proceso estándar de máquinas sociales en el π-cálculo, $M_\pi\langle w\rangle$, es que $M\langle w, \lambda_0\rangle$ no envía ningún output por el canal de salida λ_0. La incorporación de este formato previene una secuencia de reducciones posiblemente sin término entre dos máquinas. Aunque en estricto rigor podemos expresar una contribución como una cierta secuencia de reducciones, y olvidarnos de las otras *posibles* secuencias de reducciones, es importante no admitir este tipo de comportamiento a la hora de modelar máquinas reales

Con las nociones introducidas podemos expresar una contribución indirecta de la siguiente forma. Sea $\mathcal{U} = (\Gamma, \mathcal{A}, \mathcal{M}, \mathcal{C})$ una unidad de interacción social y g una pauta de interacción social. Sean N y M máquinas sociales tales que $\Lambda_N \cap \Lambda_M = \{\lambda\}$, y redefínase $g(N)$ cómo $g(N) = N\langle\varepsilon, \lambda\rangle$. Asumiremos que todas las definiciones que hemos visto a partir de la Definición 5.48 de estado de interacción social, siguen siendo coherentes a pesar de haber asignado un formato no-estandar a N. Esto es cierto, aunque no nos vamos a detener en ello. Entonces podemos expresar una contribución indirecta como un ciclo indefinido de acciones de la siguiente forma:

$$
\begin{array}{ll}
a, b \looparrowright M & \lambda_i :: w \vDash w'[w_i, w_o] \\
M \looparrowright a, b & \lambda_a, \lambda_b :: w_a, w_b \\
M \looparrowright N & \lambda :: u \vDash u'[v_i, v_o] \\
u \neq u' & \\
v_i = \alpha\, w_i\, \beta\, w_o\, \gamma & \alpha, \beta, \gamma \in (\Sigma - \{\#\})^*
\end{array}
\qquad \text{(contribución indirecta)}
$$

Donde $\mathcal{C}(\lambda_a) = a, \mathcal{C}(\lambda_b) = b$ y $\lambda_i \in \{\lambda_a, \lambda_b\}$. En la primera acción uno de los dos agentes envía un mensaje a la máquina, la que luego, en la segunda acción envía los outputs respectivos a cada agente. Luego, se produce una acción máquina-máquina de M a N con lo que N recibe un input con la información del primer par de interacción; el ciclo puede repetirse indefinidamente. Por ejemplo, supongamos que a y b son usuarios que están jugando un juego tipo GWAP. La máquina M podría haber mostrado previamente al actor a la imagen de un gato, y el juego podría consistir en que a debe describirle la imagen a b para que éste la adivine, sin nombrarle la palabra gato ni posiblemente un conjunto de palabras similares. Así, en cada ciclo, a y b van turnando el envío de inputs que ocurre en la primera acción, ingresando alternadamente una *etiqueta*, seguida de una *adivinanza*, seguida una etiqueta, etc., hasta que b adivina la imagen. Por su parte N, quien está al tanto de toda la interacción, usa este tipo de contribución para perfeccionar un sistema de etiquetado de imágenes.

El último tipo de contribución corresponde a la contribución obligada. Supongamos que M es la máquina social a la que queremos acceder, y R es la máquina a la nos vemos obligados a contribuir para lograrlo. Podemos expresar este tipo de contribuciones imponiendo condiciones sobre las reducciones, como sigue.

> *Sea a un agente cualquiera. Para toda acción $a \looparrowright M$ $\lambda_1 :: w \vDash w'[w_i, w_o]$ que comienza en un cierto estado de interacción E_1 y termina en un cierto estado de interacción E_2, existe una acción anterior $a \looparrowright R$ $\lambda_2 :: u \vDash u'[v_i, v_o]$ con $u \neq u'$ que comienza en un cierto estado de interacción E_0, pasa a un estado de interacción E_0' y termina en el estado de interacción E_1. Además se tiene para todo $D, D', D'' \in \mathcal{E}$, $D \to_{\mathcal{E}} E_1 \Rightarrow D = E_0'$ y $D'' \to_{\mathcal{E}} D' \to_{\mathcal{E}} E_1 \Rightarrow D'' = E_0$.*
>
> (contribución obligada)

La propiedad anterior establece que si un estado de interacción E_1 puede reducirse de forma de describir una acción persona-máquina sobre la máquina M, entonces existe otro estado que en dos reducciones alcanza E_1, describiendo una acción persona-máquina sobre la máquina R. Más aún, la única reducción posible anterior a E_1 es la acción persona-máquina sobre R. De esta forma no existe otra forma de reducir el estado inicial de interacción social E_g a E_1 sin hacer inmediatamente antes una contribución obligada.

Con esto damos término a la presentación del modelo formal de máquinas sociales elaborado durante la memoria.

Conclusión

Esta memoria tuvo como objetivo general proponer una formalización matemática para la noción de máquinas sociales. El primer objetivo específico consistía en estudiar las principales conceptualizaciones dentro de la literatura sobre la noción de máquinas sociales. Así, la primera parte de este trabajo estuvo dedicada completamente a analizar la noción de máquinas sociales, examinando para ello la conceptualización original de Berners-Lee, revisando luego los fenómenos más relevantes que contextualizan este concepto y estudiando finalmente los principales enfoques académicos que hasta la fecha se han desarrollado sobre máquinas sociales. El segundo objetivo específico que se planteó al comienzo de la memoria fue llevar a cabo una revisión de modelos de computación e interacción que se fueran mostrando pertinentes y adecuados para la formalización de máquinas sociales. Durante el período de tiempo en que fue realizado este trabajo se estudiaron y analizaron distintos modelos de computación e interacción que fueron evaluados para determinar cómo podrían aportar a formalizar la noción de máquinas sociales, ya fuera para usarlos directamente como base formal en el modelo que se iba desarrollando, o bien por la intuición y enfoques propios que pudieran rescatarse como interesantes de incluir en el modelo. En este sentido se estudiaron los modelos de máquinas de Turing, λ-cálculo, π-cálculo, cálculo de ambientes, máquinas de Turing persistentes, distintas teorías de computación interactiva y elementos básicos de la teoría de hiperconjuntos, entre otros. Al final el modelo de máquinas sociales incorporó explícitamente como bases formales los modelos de Turing, de máquinas de Turing persistentes y al π-cálculo. El tercer objetivo específico de esta memoria consistía en el desarrollo mismo del modelo formal de máquinas sociales, de forma que se ubicara correctamente dentro de los límites del concepto intuitivo de máquinas sociales que revisamos en la primera parte de este trabajo. Así, en la segunda parte del presente escrito nos concentramos en presentar el modelo formal de máquinas sociales que permitiera dar cuenta de las distintas nociones integran el estudio de máquinas sociales. Finalmente el último objetivo específico era evidenciar cómo el modelo desarrollado permitiría describir características prácticas de máquinas sociales reales a través de la expresión de propiedades formales que las modelasen. El final de la última parte del trabajo escrito estuvo destinado a cumplir este propósito, donde se formalizaron las características de máquinas sociales de consumo, producción, prosumo, contribución directa, contribución indirecta y contribución obligada. De esta forma se concluye que cada uno de los objetivos específicos fue cumplido, así como también el objetivo general que fundamentó este trabajo de título.

El modelo de máquinas sociales que se desarrolló se presentó en tres etapas distintas; cada una de estas etapas dio cuenta de un aspecto específico dentro del comportamiento de máquinas sociales que se quería modelar. La primera sección del modelo estuvo destinada al

comportamiento interno de la máquina, y para ello el modelo base sobre el cual se construyó nuestro formalismo fue el de máquina de Turing. En la segunda sección vimos cómo el incorporar la noción de persistencia resultó fundamental para poder describir adecuadamente a las máquinas sociales, adoptando para ello el enfoque de máquinas de Turing persistentes propuesto por Goldin et al. (2010). En la tercera sección nos preocupamos de modelar el comportamiento externo de máquinas sociales, y para ello tuvimos que ampliar la mirada y considerar distintos actores que participaban dentro de un mismo escenario de interacción. Para modelar estas interacciones nos basamos en el formalismo del π-cálculo. Cada uno de estas bases teóricas aportó de determinada forma a la construcción del modelo. El formalismo de máquinas de Turing resultó ser muy adecuado para modelar la máquina social desde el punto de vista de dispositivo computacional y de cómo se computa. En realidad el modelo de Turing entrega herramientas poderosas para poder especificar las ideas más intuitivas de etapa de computación y de paso de una etapa a otra. Luego, y como se detalló durante la misma presentación del modelo, la noción de persistencia presente en la teoría de máquinas de Turing persistentes es fundamental para modelar correctamente el comportamiento de máquinas sociales. En particular este enfoque permite formalizar la idea de un dispositivo computacional con resultados de cómputo que dependen del historial de interacciones previas con un entorno. Es impensable no considerar este comportamiento u otro que le equivalga si se quiere hacer un modelo de máquinas sociales que responda a la idea intuitiva de máquinas sociales que se comparte en la literatura. Finalmente el π-cálculo permitió modelar las interacciones entre actores lo que significó poder no sólo pensar la máquina social como un objeto matemático aislado sino también como partícipe del escenario más global donde estas máquinas habitan. La noción original de máquinas sociales es clara en establecer que estas máquinas tiene un propósito manifiesto, a saber, asistir procesos sociales dejándoles a las personas las tareas creativas y a sí mismas las tareas de administración y cómputo. En este sentido modelar la máquina social viéndola como un objeto más dentro de un entorno de interacción permitió dar cuenta de la real naturaleza que se esconde tras el concepto de máquinas sociales. El π-cálculo aportó enormemente en esto. La contribución fundamental estuvo en permitir formalizar la idea de comunicación entre actores. Al basarnos en el modelo de Turing, pudimos describir exactamente cómo se realizan las computaciones, por lo tanto lo que nos faltaba para describir completamente las interacciones entre personas y máquinas era definir cómo una persona o máquina social le envía un input a otra máquina, y cómo ésta le retorna un output al emisor original. Esto lo logramos modelar con el π-cálculo. Por último, otra contribución clave del π-cálculo al modelo fue que permite introducir la idea de canal de comunicación. Este es un punto muy interesante, puesto que el enfoque de máquinas de Turing persistentes, que extiende el formalismo de Turing para poder modelar interacciones con un entorno, no es suficiente para permitir a la máquina distinguir quién está enviando los mensajes. Este enfoque condensa todos los actores que componen el entorno bajo la identidad de un sólo actor: el entorno mismo. Dicho de otro modo, al terminar la primera sección contábamos formalmente con un dispositivo computacional que respondía con el mismo output ante el mismo input, y no daba cuenta de interacciones con un entorno. Al terminar la segunda sección, contábamos formalmente con una máquina que almacenaba un estado entre computaciones que se realizaban al interactuar con un entorno, lo que le permitía poder retornar outputs que fueran dependientes de esas interacciones. Sin embargo, la máquina no sabía quién enviaba esos mensajes, ni quién los recibía, y nuestro modelo tampoco podía dar cuenta de cómo estas interacciones afectaban al entorno. Finalmente, gracias al π-cálculo, pudimos incluir las ideas de comunicación entre actores, con lo que la máquina social, vista

ahora como un proceso del π-cálculo con un conjunto específico de canales de comunicación, puede saber efectivamente con quién se comunica y el modelo final puede describir cómo se suceden las distintas interacciones entre los actores que participan del entorno.

Una de las tareas futuras que deja este trabajo tiene que ver con un análisis sobre los efectos y consecuencias que conlleva el haber incluido dos visiones o modelos de computación de naturaleza diferente. Mientras que las máquinas de Turing y las máquinas de Turing persistentes pueden considerarse como parte un mismo tipo de formalismos, el π-cálculo tiene una motivación, un fundamento teórico, y una forma de proceder distintos. Uno de los puntos importantes a considerar en el futuro de esta investigación es si realmente es necesario partir desde dos bases formales distintas o puede construirse el modelo sólo en base a uno de los formalismos. Este análisis es muy importante, y también da luces de ser bastante complejo puesto que se da en un espacio metateórico. En una primera aproximación, puede decirse que los resultados sobre equivalencia de expresividad entre formalismos, al estilo de lo que sucede entre máquinas de Turing y el λ-cálculo, no son criterios que tengan un mayor peso en el análisis. El π-cálculo ya admite una codificación del λ-cálculo y con ello es igual de expresivo que los modelos clásicos de computación. Los criterios que parecen ser más relevantes a primera vista pueden agruparse en tres áreas. La primera tiene que ver con el estudio de las diferencias fundamentales entre modelar un concepto en términos de funciones de transición y sucesión de configuraciones, versus modelarlo en términos de reducciones. La segunda área tiene que ver con la necesidad de considerar fenómenos de naturaleza circular, donde el concepto de streams parece jugar un rol clave. Aquí, el punto es que la axiomática de conjuntos en que se fundamentan las máquinas de Turing persistentes es un elemento que no participa habitualmente en las discusiones del modelo original de Turing, y pareciera ser una restricción sobre el tipo de teoría de conjuntos sobre el que puede construirse el formalismo de Turing si se quiere incorporar la persistencia completamente. Ahora bien, en el caso del π-cálculo, la discusión es si el proceso que representa el comportamiento de una máquina social se reduce finalmente al proceso inerte o no, es decir, termina o no en algún momento. Si se escoge modelar la máquina social como un proceso que no tiene término, como se hizo en el modelo que se elaboró en esta memoria, entonces la circularidad del fenómeno que se modela pareciera poder suplirse con la inclusión de la definición paramétrica. Sin embargo, reconstruir la noción de lenguaje de stream persistente sin que esto termine siendo un mero renombre del concepto de stream y la teoría que lo sustenta aparece como un problema realmente complejo de resolver. Finalmente la tercera área de análisis que surge a primera vista tiene que ver con la posibilidad de extender el formalismo de Turing trabajando las cintas como canales a la vez, para así obtener los beneficios que el π-cálculo otorga al modelo de máquinas sociales. Por supuesto todas estas áreas de análisis requieren una revisión muy delicada.

Para cerrar este capítulo de conclusiones, y con ello dar fin a este documento de trabajo de título, quisiera expresar mi opinión personal como estudiante sobre uno de los temas que más me ha llamado la atención del concepto que he estado estudiando el último tiempo. El grupo de fenómenos de la vida real que se intentan estudiar bajo la noción de máquinas sociales son también estudiados, como sabemos, bajo una serie de nociones en principio distintas entre sí y que se diferencian por el enfoque y la aproximación que realizan. La noción de máquinas sociales es una más dentro de un grupo de nociones con una base común. Es importante haber puesto en su contexto la noción de máquinas sociales, situándola al lado

de aquellas que revisamos en la primera parte de la memoria. Desde mi punto de vista, estos conceptos pueden clasificarse en dos grupos fundamentalmente distintos. En el primero, se estudia la inteligencia humana y la capacidad de entendernos como seres sociales. Pienso que cuando se hace alusión a la inteligencia colectiva implícita en Google o Wikipedia, y se intentan establecer elementos de juicio para diferenciar los distintos tipos de inteligencia colectiva que cada uno representan, el desarrollo informático y computacional solo juega un rol de catalizador de procesos sociales. Creo que en ningún caso estas nuevas formas y/o manifestaciones de inteligencia son únicas ni exclusivas de la computación, sino más bien de todo salto tecnológico de la categoría del que implicó la aparición del computador, el Internet y la Web. Así, tenemos un grupo de conceptos bajo los cuales se estudia este panorama intelectual. Ahora bien, el tema al que me refería al comienzo de este párrafo y que más me ha llamado la atención al desarrollar la memoria, es que en el otro grupo parece no haber una diferenciación clara entre los distintos conceptos que estudian cómo se funden los elementos sociales y computacionales en el software. La imagen que me termino formando de la noción de máquinas sociales, es que es realmente muy similar a la de *crowdsourcing* y otras que estudié durante el transcurso de la memoria. Para ser preciso, voy a dar un ejemplo. En el área de *crowdsourcing* el trabajo de Doan et al. (2011) —que analizamos en la primera parte de la memoria— es uno de lo más relevantes del último tiempo. En el área de máquinas sociales, se tiene la misma situación con el trabajo de Shadbolt et al. (2013). El primero de estos trabajos desarrolla una categorización de las principales propiedades exhibidas por los sistemas de *crowdsourcing*. El segundo desarrolla un framework para clasificar máquinas sociales en base a los constructos que resulten más apropiados para ello. Cada trabajo define el concepto que estudia y en principio esas definiciones son claramente distintas. Cada trabajo hace una caracterización de su propio objeto de estudio y de las principales nociones que intervienen en ese estudio. Es decir, estos trabajos están exhibiendo el enfoque particular con que la noción de *crowdsourcing* por un lado y la de máquina social por el otro estudian la mezcla de elementos sociales y computacionales. Sin embargo, si miramos los resultados de estos trabajos —presentados en la Sección 2.5 y Sección 3.4, respectivamente— nos damos cuenta inmediatamente que en su mayoría las propiedades propuestas por Doan et al. son también constructos elicitados por Shadbolt et al. Es decir los enfoques con que se estudian estos conceptos son muy similares. De todo esto concluyo que es mejor abstraerse de un intento por definir cada concepto como una noción precisa y estable en el tiempo y entre autores, y en cambio acercarse a la noción de máquinas sociales entendiendo el contexto que la rodea para recoger desde los distintos enfoques de máquinas sociales y desde ese contexto, las ideas más interesantes y novedosas para el estudio del fenómeno base que se intenta explicar. Así, el modelo de máquinas sociales que se propuso en esta memoria buscó dar un trato formal a ese fenómeno original que fundamenta el estudio de máquinas sociales.

Bibliografía

Amazon Mechanical Turk. (s.f.). *Working on HITs.* `https://www.mturk.com/mturk/welcome?variant=worker`. ([Recuperado el 11 de Agosto de 2014])

Barwise, J., y Moss, L. (1996). *Vicious circles: on the mathematics of non-wellfounded phenomena*. Stanford, California: Center for the Study of Language and Information.

Berners-Lee, T. (1997). *Realising the full potential of the web.* `http://www.w3.org/1998/02/Potential.html`. ([Recuperado el 7 de Agosto de 2014])

Berners-Lee, T. (1998). *The world wide web: A very short personal history.* `http://www.w3.org/People/Berners-Lee/ShortHistory.html`. ([Recuperado el 7 de Agosto de 2014])

Berners-Lee, T., y Fischetti, M. (2000). *Weaving the web: The original design and ultimate destiny of the world wide web by its inventor*. New York, NY: HarperCollins.

Buregio, V., Meira, S., y Rosa, N. (2013). Social machines: a unified paradigm to describe social web-oriented systems. En *Proceedings of the 22nd international conference on world wide web companion* (pp. 885–890).

Doan, A., Ramakrishnan, R., y Halevy, A. Y. (2011). Crowdsourcing systems on the world-wide web. *Communications of the ACM*, *54*(4), 86–96.

Goldin, D. Q., Smolka, S. A., Attie, P. C., y Sonderegger, E. L. (2004). Turing machines, transition systems, and interaction. *Information and Computation*, *194*(2), 101–128.

Hendler, J., Shadbolt, N., Hall, W., Berners-Lee, T., y Weitzner, D. (2008). Web science: an interdisciplinary approach to understanding the web. *Communications of the ACM*, *51*(7), 60–69.

Hetmank, L. (2013). Components and functions of crowdsourcing systems – a systematic literature review.

Howe, J. (2006). The rise of crowdsourcing. *Wired magazine*, *14*(6), 1–4.

Howe, J. (2008). *Crowdsourcing: How the power of the crowd is driving the future of business*. New York, NY: Random House.

Howe, J. (s.f.). *Crowdsourcing.* `http://www.crowdsourcing.com/`. ([Recuperado el 11 de Agosto de 2014])

Klein, M. (2008). Achieving collective intelligence via large-scale argumentation. *Collective Intelligence: Creating a Prosperous World at Peace, Earth Intelligence Network, Oakton, Virginia*, 475–484.

Leimeister, J. M. (2010). Collective intelligence. *Business & Information Systems Engineering*, *2*(4), 245–248.

Lewis, H. R., y Papadimitriou, C. H. (1981). *Elements of the theory of computation.* Englewood Cliffs, New Jersey: Prentice-Hall.

Malone, T. W. (2008). What is collective intelligence and what will we do about it? *Collective Intelligence: Creating a Prosperous World at Peace, Earth Intelligence Network, Oakton, Virginia*, 1–4.

Meira, S. R. L., Buregio, V. A. A., Nascimento, L. M., de Figueiredo, E. G. M., Neto, M., Encarnação, B. P., y Garcia, V. (2010). *The emerging web of social machines* (Inf. Téc.).

Milner, R. (1990). Functions as processes. En *Proceedings of the 17th international colloquium on automata, languages and programming* (pp. 167–180).

Milner, R. (1999). *Communicating and mobile systems: the pi-calculus.* Cambridge, Inglaterra: Cambridge University Press.

Noubel, J.-F. (2008). Collective intelligence: From pyramidal to global. *Collective Intelligence: Creating a Prosperous World at Peace, Earth Intelligence Network, Oakton, Virginia*, 225–234.

Pierce, B. C. (1997). Foundational calculi for programming languages. *The Computer Science and Engineering Handbook*, *1997*, 2190–2207.

Rheingold, H. (2003). *The next revolution: Smart mobs.* `http://www.openp2p.com/lpt/a/3295`. ([Recuperado el 24 de Abril de 2014])

Rheingold, H. (2004). Network logic: Who governs in an interconnected world? En H. McCarthy, P. Miller, y P. Skidmore (Eds.), (pp. 191–202). Londres, Inglaterra: Demos.

Ritzer, G., Dean, P., y Jurgenson, N. (2012). The coming of age of the prosumer. *American Behavioral Scientist*, *56*(4), 379–398.

Shadbolt, N. (2013). Knowledge acquisition and the rise of social machines. *International Journal of Human-Computer Studies*, *71*(2), 200–205.

Shadbolt, N. R., Smith, D. A., Simperl, E., Van Kleek, M., Yang, Y., y Hall, W. (2013). Towards a classification framework for social machines. En *Proceedings of the 22nd international conference on world wide web companion* (pp. 905–912).

Surowiecki, J. (2005). *The wisdom of crowds*. New York, NY: Anchor Books.